DOCTOR OF PHILOSOPHY

Quality of service for high-speed interconnection networks onboard spacecraft

Ferrer Florit, Albert

Award date:
2013

DOCTOR OF PHILOSOPHY

Quality of service for high-speed interconnection networks onboard spacecraft

Albert Ferrer Florit

2013

University of Dundee

Quality of Service for High-Speed Interconnection Networks Onboard Spacecraft

A. Ferrer-Florit

Quality of Service for High-Speed Interconnection Networks Onboard Spacecraft

Albert Ferrer-Florit

Doctor of Philosophy

University of Dundee

November 2013

Table of Contents

Table of Figures

Table of Tables

Acknowledgments

The author wishes to thank several people for helping, assisting and providing support during the process of doing this work and writing this thesis.

First, I would like to thank Steve Parkes and Martin Suess who gave me the opportunity to perform this research in a topic which applies my passion for engineering challenges to my motivation for the everlasting frontier, Space. Also, this work would not have been possible without the support of ESA's Networking/Partnering Initiative (NPI).

A special gratitude I give to Steve Parkes, for his patience, for the time which he has spent assisting my research, and for his excellent supervision. I would also like to thank the rest of the Space Group at the University of Dundee, and the people in TEC-ED at ESTEC for all the assistance they have provided. I would like to especially thank Alberto Gonzalez for his work related with this thesis during his stay in ESTEC.

I am also grateful to the people in SpaceWire Working Groups and SpaceWire conferences who have participated in technical discussions and have provided feedback related with my research.

I cannot thank my parents enough for their endless love, patience and support. They have helped me to settle far from home and have provided useful advice when needed. I would also like to thank my partner, Neus, for her love and kindness and the sacrifices she has taken during this time.

Finally, I would also like to thank my friends and family for trying to show interest when talking passionately about technical topics and for lobbying with everybody else for the completion of this thesis.

Declaration of the Candidate

I hereby declare that I am the author of this thesis; that the work described in the thesis is my own; that it has not previously been put forward in submission for any other degree or qualification; and that I have consulted the table of references herein.

Albert Ferrer-Florit

November 2013

Declaration of the Supervisor

I declare that Albert Ferrer-Florit has satisfied all the terms and conditions of the regulations made under Ordinances 12 and 39, and has completed the required nine terms of research to qualify in submitting this thesis in application for the degree of Doctor of Philosophy.

Prof. S. M. Parkes

November 2013

Abstract

State-of-the-art onboard spacecraft avionics use SpaceWire networks to interconnect payload data-handling sub-systems. This includes high data-rate sensors and instruments, processing units, and memory devices. SpaceWire is an interconnection network composed of nodes and routers connected by bi-directional, point-to-point, high-speed, serial-data communication links. SpaceWire is established as one of the main data-handling protocols and is being used on many ESA, NASA and JAXA spacecraft.

SpaceWire is very successful for being fast, flexible and simple to use and implement. However it does not implement Quality of Service mechanisms, which aim to provide guarantees in terms of reliability and timely delivery to data generated by network clients. Quality of Service is increasingly being deployed in commercial ground technologies and its availability for space applications, which requires high reliability and performance, is of high interest for the space community.

This thesis researches how Quality of Service can be provided to existing SpaceWire networks. Existing solutions for ground-based technologies cannot be directly used because of the constraints imposed by the limitations of space-qualified electronics. Due to these limitations SpaceWire uses wormhole routing which has many benefits but makes it more challenging to obtain timing guarantees and to achieve a deterministic behaviour.

These challenges are addressed in this work with a careful analysis of existing Quality of Service techniques and the implementation of a novel set of protocols specifically designed for SpaceWire networks. These new protocols target specific use cases and utilise different mechanisms to achieve the required reliability, timely delivery and determinism. Traditional and novel techniques are deployed for first time in SpaceWire

networks. In particular, segmentation, acknowledgements, retry, time-division multiplexing an cross-layer techniques are considered, analysed, implemented and evaluated with extensive prototyping efforts.

SpaceWire provides high-rate data transfers but the next generation of payload instruments are going to require multi-gigabit capabilities. SpaceFibre is a new onboard networking technology under development which aims to satisfy these new requirements, keeping compatibility with SpaceWire user-applications. As a new standard, SpaceFibre offers the opportunity to implement Quality of Service techniques without the limitations imposed by the SpaceWire standard.

The last part of this thesis focuses on the specification of the SpaceFibre standard in order to provide the Quality of Service required by next generation of space applications. This work includes analytical studies, software simulations, and hardware prototyping of new concepts which are the basis of the Quality of Service mechanisms defined in the new SpaceFibre standard. Therefore, a critical contribution is made to the definition and evaluation of a novel Quality of Service solution which provides high reliability, bandwidth reservation, priority and deterministic delivery to SpaceFibre links.

Chapter 1: Introduction

1.1 Background

The current generation of spacecraft avionics has to deal with the increasing high data volume produced by onboard instruments and related payload data processing units. For this reason, in payload data-handling, low data rate serial buses such as MIL-STD-1553B and CAN bus have been replaced in most cases by SpaceWire networks [Parkes 1999], [ECSS 2003b], [ECSS 2008].

SpaceWire is a standard for interconnection networks designed to connect together high data-rate sensors, processing units, memory devices and other sub-systems onboard spacecraft. It provides high-speed (2 to 200 Mbits/s), bi-directional, full-duplex data links which connect together SpaceWire enabled equipment. Networks can be built to suit particular applications using point-to-point data links and routing switches. The main function of a SpaceWire routing switch, or router, is to forward SpaceWire packets by establishing non-conflicting connections between input and output ports of the router.

Missions using SpaceWire include Bepi Colombo, Earthcare and GAIA from ESA; James Webb Space Telescope, Lunar Reconnaissance Orbiter, GOES-RT and SWIFT from NASA; and Bepi Colombo MMO and NeXT from JAXA. There are many other industrial developments of chips, sub-systems and space missions using SpaceWire [Parkes 2012b].

SpaceWire is very successful for being fast, flexible and simple to use and implement. The price is that the transport layer of the OSI model [ITU 1994] is not included as SpaceWire only provides the physical, data link and network layer (Figure 1-1). The SpaceWire network layer provides basic packet delivery with best effort service.

An additional transport layer protocol is required to obtain end-to-end communication services with Quality of Service.

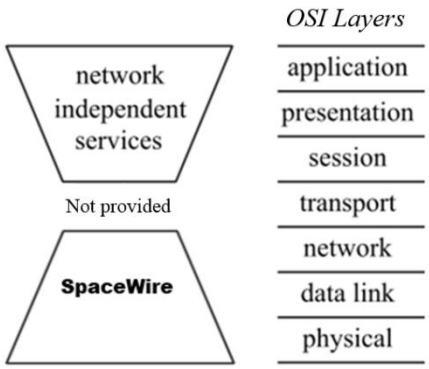


Figure 1-1: SpaceWire in the context of OSI model

Quality of Service (QoS) is a set of techniques to manage network resources in order to provide different priorities to different applications or to guarantee a certain level of performance to a packet flow, which is a sequence of packets with the same QoS sent from a particular source to a particular destination.

Quality of Service control mechanisms can provide guarantees in terms of data rate (throughput), delay (latency), or error rate. This allows the decoupling of the application or network client from the interconnection network so they can be designed independently from each other [Keutzer 2000].

Some QoS mechanisms can be provided in the transport layer. For example, reliability can be provided on top of an unreliable network service using data acknowledgments and data retransmissions. However, time-related guarantees can be very difficult to provide if the lower layers do not offer these guarantees. An interesting solution is to use cross-layer techniques that apply QoS mechanisms across multiple layers of the OSI model [Shakkottai 2003]. Another obvious solution is to modify some aspects of the underlying network layers. This second approach is the one adopted by SpaceFibre technology.

SpaceFibre is an emerging standard that provides gigabit data rates to satellites and spacecraft to meet the increasing demands of ever-more sophisticated technological advances of onboard instrumentation [Parkes 2009]. SpaceFibre is compatible with applications that use SpaceWire to send packets because it uses the same network layer. The new link layer uses 8b10b encoding to achieve gigabit rates and it uses the concept of virtual channels to allow the implementation of efficient QoS control mechanisms.

1.2 Description of Problem

Space applications require a communication architecture with a high degree of reliability and availability, which must be achieved with limited resources. The space environment, in particular, radiation effects, impose serious limitations on the use of electronic components, so the space qualified devices are usually several generations behind the state-of-the-art ground terrestrial technology [Dier 2002]. This makes the ground-based commercial protocols very difficult to use in space even if they provide the Quality of Service required. Therefore, the technologies developed specifically for space, like SpaceWire, are very successful in space applications, because they take into account the limitations of space-qualified electronic technology (i.e. limited gate count in radiation tolerant chips).

SpaceWire was designed to meet the demands for high data rate payload instruments but does not provide the reliability and deterministic delivery required for some command and control operations. This means that other avionics sub systems, including some payload data-handling architectures, use a dedicated low-speed control serial bus for this function, which adds complexity and increases the cost of the overall avionics system.

The main space agencies represented within the Consultative Committee for Space Data Standards (CCSDS) and the SpaceWire Working Group have highlighted the need for the

development of a set of protocols that can provide the Quality of Service required using existing SpaceWire technology and considerable work has already been done [Parkes 2008] [GSFC 2005].

SpaceWire networks use Wormhole Switching to achieve the highest performance with minimum hardware resources. With this technique, packets are routed immediately when the first bytes containing the destination address are received in the router, so a single packet can occupy multiple consecutive routers and links, like a worm. This reduces latency significantly compared to other switching techniques like store-and-forward used by Ethernet and other popular networking technologies [McKinley 1993].

However, wormhole switching makes the network more susceptible to congestion as a blocked packet occupies multiple links causing other packets to block, so the blocking spreads across the network. It is therefore necessary to develop and test different mechanisms that alleviate or solve this problem in order to provide timeliness related Quality of Service [Gerla 1996]. These mechanisms have to be compatible with existing SpaceWire devices and protocols to minimise the cost of their deployment.

SpaceFibre also uses wormhole switching at network layer but it introduces virtual channels at the link layer. Virtual channels are multiplexed over a link, so in case a packet using a virtual channel experiences congestion, it does not affect other packets using other virtual channels over the same link. However, the arbitration between virtual channels needs to be carefully defined in order to provide a high degree of QoS with priorities, guaranteed throughput, and determinism.

Finally, the reliability aspects of Quality of Service, i.e. FDIR, needs also to be traded off for both SpaceWire and SpaceFibre with the goal of using as little resources as possible. In SpaceWire, reliability should be implemented at the transport layer, as the link error

rate is very low. In SpaceFibre, its very high data rate implies that it is more likely to have an error so the reliability has to be provided at link layer. It is a good practice to detect and recover from errors as close as possible to the source of the error [Dally 2003].

1.3 Scope and objectives

This thesis will address the implementation of Quality of Service techniques to high speed interconnection networks based on SpaceWire and SpaceFibre links. SpaceWire is a mature technology, so no changes to the SpaceWire standard will be considered. Therefore the solutions proposed will be compatible with existing devices and will cover multiple use cases with different combinations of the following requirements:

- Reliable point-to-point or end-to-end connections.
- Synchronous and asynchronous interconnection networks.
- Unidirectional and bidirectional user data flows including asymmetric data rates.
- Strict guarantees in latency and throughput metrics.
- Sporadic, periodic or constant data sources.
- Support for high data rate payload traffic.
- Support for command and control traffic.

For SpaceFibre links, the Quality of Service requirements of the draft specification will be evaluated and the proposed solutions implemented with real hardware, so a complete specification can be provided.

In order to achieve these objectives, the last techniques used in other application domains, such as cross-layer solutions will be evaluated for their used in SpaceWire and SpaceFibre links.

1.4 Research Questions

The research questions that arise from the previous discussion are the following.

- How should a Quality of Service layer be implemented for SpaceWire?
- Is it useful to use Cross-Layer Quality of Service techniques over SpaceWire?
- How should the new SpaceFibre protocol be designed to provide the required Quality of Service?

It is the aim of this thesis to answer these questions.

1.5 Publications and other Achievements

Appendix A lists the publications of the author of this thesis, who has participated in the following projects:

- "SpaceNet": European Space Agency (ESA), contract No. 220774-07-NL/LvH
- "SpaceFibre": European Space Agency (ESA), contract No.17938/03/NL/LvH
- "SpaceFibre Very High Speed Link Demonstrator": European Space Agency (ESA), contract No. 4000102641
- "SpaceWire RT": European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 263148
- "Very High Speed Serial Interfaces": European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 284389

The author has actively participated in the SpaceWire Working Group meetings, with multiple presentations and on site demonstrations of the studies described in the following chapters. This has been very useful for gathering feedback from the SpaceWire community and providing inputs to other related projects.

Most important, the work described in this thesis has made a critical contribution to the definition of the following protocols that are expected to be an ECSS standard in the near future:

- SpaceWire-D protocol: Prototyping and analysis of its deterministic QoS.
- SpaceFibre protocol: Definition, simulation and prototyping of the retry mechanism. Significant contribution to the definition, simulation and prototyping of the arbitration mechanism required to provide QoS to the SpaceFibre Virtual Channels (VC).

1.6 Structure of Thesis

Following this introduction, a background chapter describes state-of-the-art technologies used onboard spacecraft, in particular, SpaceWire. It also explains fundamental Quality of Service concepts and show how they are applied in avionics and other application domains.

After this background information, a short chapter is provided describing the research questions in greater detail, their importance, and how they will be answered in this thesis.

The fourth chapter analyses different Quality of Service control mechanisms and which ones are best suited to SpaceWire networks, performing simulations and experiments when required. These results are the basis of the protocols developed in the following two chapters.

Chapter 5 aims to answer the first research question, "How should a Quality of Service layer be implemented for SpaceWire?", with the definition, prototyping and evaluation of two protocols with QoS. The first is a bidirectional general purpose protocol and the second is specific for unidirectional user data flows in scheduled networks.

Chapter 6 aims to answer the second research question, "Is it useful to use Cross-Layer Quality of Service techniques over SpaceWire?", with the definition, prototyping and evaluation of different protocols that try to provide similar or better QoS metrics with less protocol overhead using cross-layer techniques.

The last research question "How should the new SpaceFibre protocol be designed to provide the required Quality of Service?" is the focus of the next chapter, which it is followed by a chapter summarizing the conclusions.

For clarity, the work is usually divided between the QoS aspects of reliability and timeliness, and the results are compared taking into account the platform used by the experiment.

Please note that there is a list of acronyms and abbreviations at the end. There is also a glossary that defines the terms used in the context of this thesis.

Chapter 2: State-of-the-art

In this chapter an overview is given first about state-of-the-art spacecraft avionics, network technologies and protocols onboard spacecraft. In particular, one of the main focus of this thesis, SpaceWire and its related technologies, is explained in detail. Generic network concepts are also introduced when it is required to understand the underlying protocols.

The next section presents Quality of Service concepts which are then reviewed in the context of current avionics platforms, including previous work related with this thesis. QoS techniques in other application domains are also briefly presented to serve as an inspiration for the implementation of QoS for SpaceWire and SpaceFibre networks.

2.1 Spacecraft Avionics

Avionics are the electronic systems used onboard aircraft and artificial satellite or spacecraft [Spitzer 2000]. The architecture of Spacecraft avionics is traditionally divided between platform and payload data-handling functions [Tramutola 2011].

Figure 2-1 shows this division with the platform side being related with operational functionality common in all spacecraft (navigation, power, temperature) and the payload side being mission specific (instruments, data downlink and mission data analysis). This scheme shows the use of different onboard communications architectures for Payload and Platform.

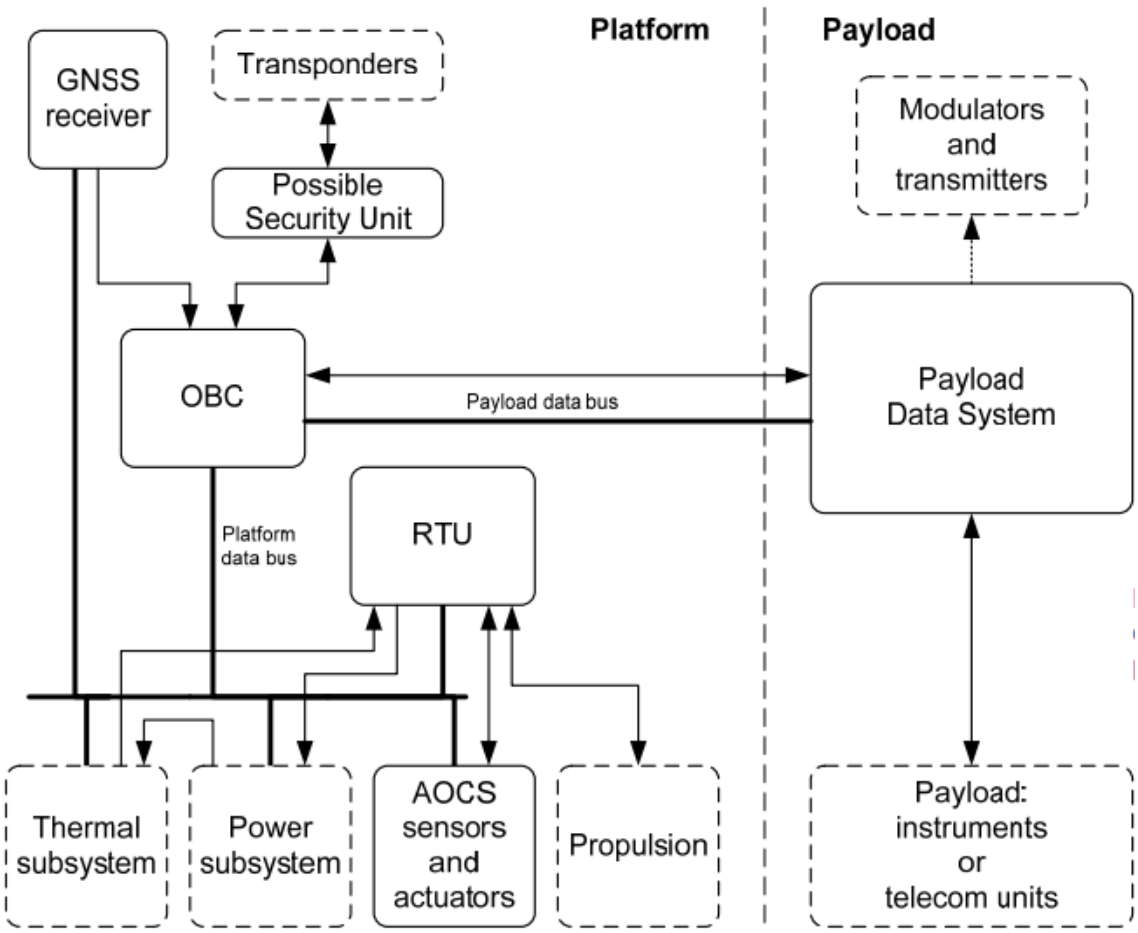


Figure 2-1: Platform and Payload spacecraft avionics [Hult 2011]

The platform data bus has borrowed the architecture from aeronautics and it is based on the use of the MIL-STD-1553 bus standard initially developed for military use [Borky 1996]. Currently, the use of this low-speed bus as a payload data bus has been replaced in most space missions by more advanced technologies, in particular, SpaceWire [Parkes 1999]. Payload onboard data-handling based on SpaceWire provides point-to-point and networked architectures at much higher data rates with less cost in terms of power and mass.

Figure 2-2 illustrates the SpaceWire data-handling architecture used on ExoMars [Dean 2008]. ExoMars is an ESA mission to Mars that incorporates a versatile rover that uses several different types of cameras to support navigation (PanCams, NavCams and

LocCams). The processing of this image data is quite intensive so a dedicated image processing chip is used to support the processing. SpaceWire is used to transfer images from the cameras to mass memory and from there to the processor and image processing chip. A SpaceWire router implemented within the OBC is used to interconnect the various SpaceWire units.

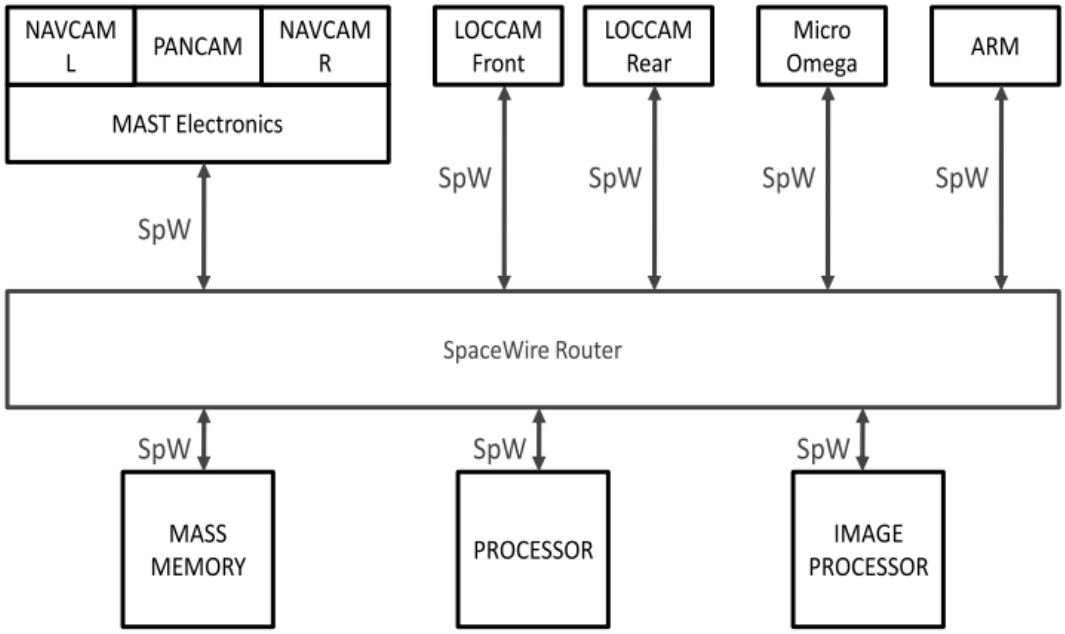


Figure 2-2: ExoMars SpaceWire Data-Handling Architecture [Parkes 2012b]

Other future ESA missions plan to use even more complex payload data-handling architectures. Figure 2-3 shows the payload data-handling section of the electrical architecture of EarthCARE, a planned space mission by the European and Japanese space agencies [ESA 2004]. EarthCARE is an acronym standing for Earth Clouds, Aerosols and Radiation Explorer so the aims of the mission are to improve understanding of the cloud, radiative and aerosol processes that affect the Earth's climate.

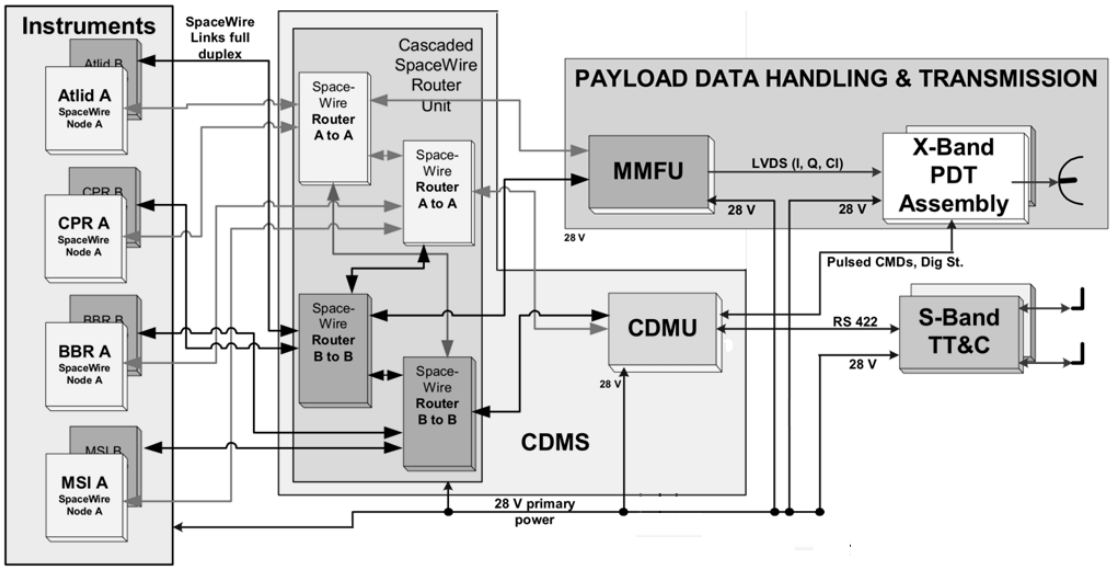


Figure 2-3: EarthCARE payload data-handling [ESA 2004]

In this scheme, all platform data-handling functions are centralised in the Command and Data Management Unit (CDMU), which also controls the payload instrument data management via SpaceWire interfaces. In case of failures, there are redundant instruments and SpaceWire routers, all of them are interconnected with SpaceWire links. The main task of the payload data-handling is to compress instrument data, store up to 302Gbytes in the Mass Memory and Formatting Unit (MMFU) and transmit this payload data to Earth via an X-band with a rate up to 150Mb/s. This is expected to be done using ERC32 or LEON radiation-tolerant processors [Habinc 2010].

2.1.1 Onboard Payload Data-Handling

In most of space applications supported by major space agencies, state-of-the-art onboard payload data-handling architectures and space avionics are driven by protocols and technologies promoted by the Consultative Committee for Space Data Standards (CCSDS). Founded in 1982 by the major space agencies of the world, the CCSDS is a multi-national forum for the development of communications and data systems standards for spaceflight. The goal is to enhance governmental and commercial interoperability and

cross-support, while also reducing risk, development time and project costs. In Europe, the European Cooperation for Space Standardization (ECSS) is responsible for the publication and maintenance of the set of standards for use in all European space activities.

The definition of a European reference architecture for onboard payload data-handling is being undertaken by the SAVOIR group, which stands for Space Avionics Open Interface Architecture and it is led by European space agencies and industries [Hjortnaes 2011].

The baseline of the proposed reference architecture is based on the use of standard building blocks and data link-layers that provide the services defined by the Spacecraft Onboard Interface Services (SOIS) via hardware or software means [CCSDS 2007].

Figure 2-4 illustrates the series of services defined by SOIS which are independent of the specific link layer used (SpaceWire, MIL-STD-1553B and CAN). The most relevant are:

- Command and Data Acquisition Services, that provide mechanisms for commanding and acquiring data from devices within a spacecraft;
- Message Transfer Service, that transfer of messages between software applications within a spacecraft.
- Packet Service, which transfers packets between data systems within a link-layer of a spacecraft.
- Memory Access Service, that provides access to memory locations of a data system from another data system within a subnetwork of a spacecraft.

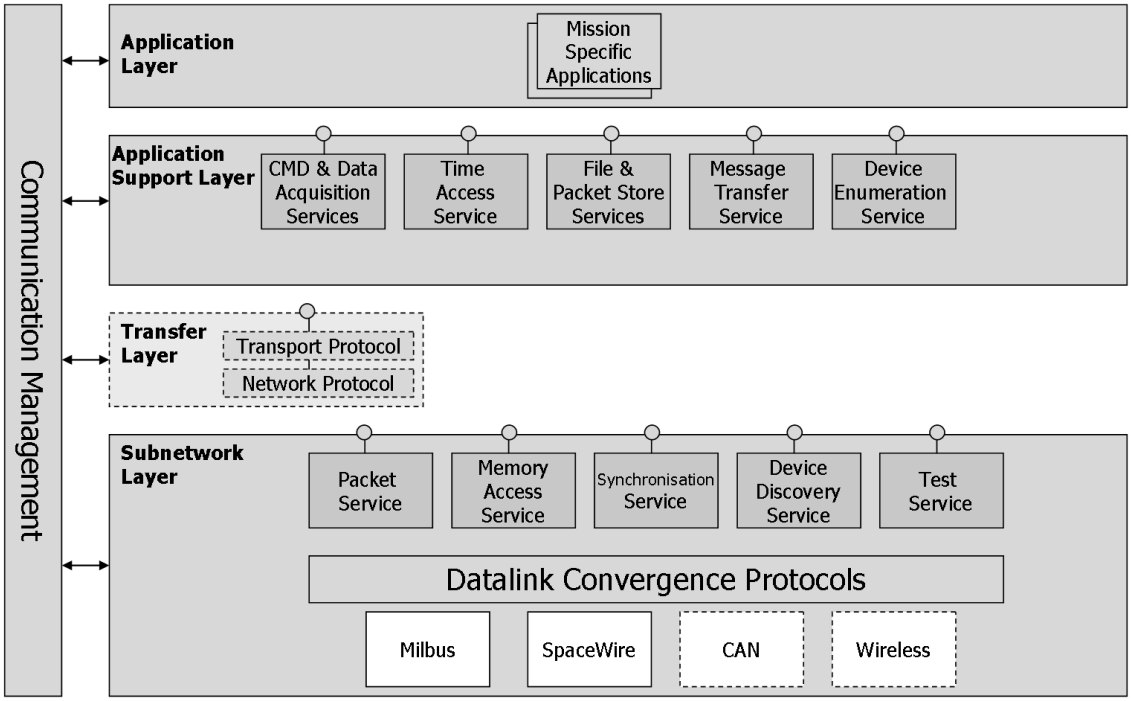


Figure 2-4: SOIS Architecture [CCSDS 2007]

Regarding how these SOIS services are used in real space applications, the Packet Utilisation Standard (PUS) defines an application-level interface between ground and space, in order to satisfy the requirements of electrical integration and testing of flight operations [ECSS 2003].

There is an ongoing effort to map these services to the data link protocols being used, i.e. SpaceWire, CAN and MIL-STD-1553B [Notebaert 2008]. For example, SpaceWire only provides natively the Packet Service. The Memory Access and Command and Data Acquisition Service can be provided using RMAP protocol [ECSS 2010b]. The Message Transfer Service requires a transport protocol that provides a certain Quality of Service. Chapter 5 of this thesis deals with the definition of these transport protocols.

2.1.1.1 Data Link Protocols

This section introduces the data link protocols and bus architectures typically used in spacecraft avionics.

	MIL-STD-1553	CAN	SpaceWire	SpaceFibre
Medium access	Time-triggered (TDM, Scheduled)	Event-triggered (with arbitration)	Event-triggered	Event-triggered & Time-triggered
Topologies	Bus	Bus	Point-to-Point, Networked	Point-to-Point, Networked
Reliability	Bus guardian	Optional	Parity bit error detection	Retry mechanism, redundant lanes.
Speed	Up to 1Mbps	Up to 1Mbps	Up to 200Mbps	2.5Gbps per lane
Target application	Platform data bus	Low speed platform & payload data bus	Payload data-handling	High data-rate Payload data-handling with QoS
Missions used	Most ESA and NASA major missions	Exomars	Most ESA and NASA new missions	Protocol under development

2.1.1.1.1 MIL-STD-1553

MIL-STD-1553 [DOD 1975] is a serial data bus that features a dual-redundant balanced-line physical layer, a differential network interface, Time-Division Multiplexing (TDM), half-duplex command/response protocol and supports up to 31 terminals devices.

A MIL-STD-1553 bus consists of a Bus Controller controlling multiple Remote Terminals that operate at a bit rate of 1Mbit/s. A terminal device cannot originate a data transfer of itself. Requests for transmission from terminal devices are handled by the controller polling the terminals. Messages consist of one or more 16-bit words (command, data or status). The bus can be made dual or triple-redundant by using several independent

wire pairs where all devices are connected to all buses. There is provision to designate a new bus control computer in the event of a failure by the current master controller.

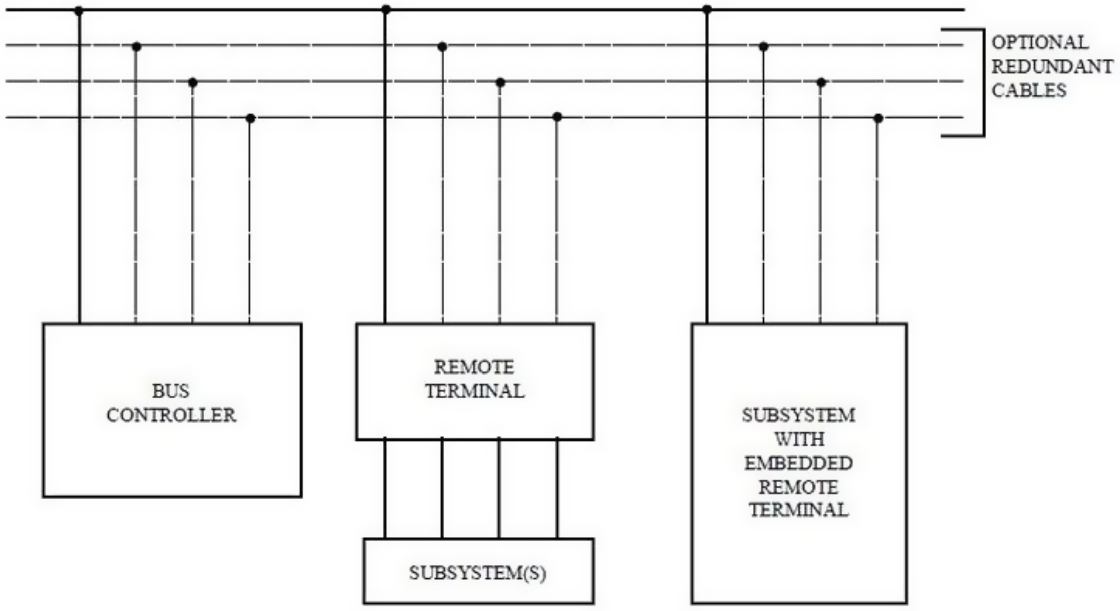


Figure 2-5: Example of a MIL-STD-1553 data bus architecture

2.1.1.1.2 CAN Bus

CAN bus (Controller Area Network) [ISO 2003] is a multi-master broadcast serial bus developed by the automotive industry. CAN features an automatic 'arbitration free' transmission: A CAN message that is transmitted with highest priority will 'win' the arbitration, and the node transmitting the lower priority message will sense this, back off and wait. Bit rates up to 1 Mbit/s are possible at network lengths below 40 m. CAN has four frame types that are transmitted serially:

- Data frame: a frame containing node data (up to 8 bytes) for transmission
- Remote frame: a frame requesting the transmission of a specific identifier
- Error frame: a frame transmitted by any node detecting an error
- Overload frame: a frame to inject a delay between data and/or remote frames

2.1.1.1.3 SpaceWire

SpaceWire [ECSS 2003b] is a bi-directional, point-to-point, high-speed serial data communication link, used for spacecraft onboard data-handling. It provides a unified high-speed data-handling infrastructure for connecting together different pieces of equipment. The basic idea was born from the IEEE-1355 standard, and it is based on an LVDS physical layer. With the leading role of the European Space Agency (ESA) SpaceWire has become a widely accepted standard suitable for space applications. It is described in detail in the next section.

2.1.1.1.4 SpaceFibre

SpaceFibre (SpFi) is a new very high-speed serial link designed specifically for use onboard spacecraft [Parkes 2007]. The aim of SpaceFibre is to provide point-to-point and networked interconnections for gigabit rate instruments, mass-memory units, processors and other equipment, onboard a spacecraft. SpaceFibre carries SpaceWire packets over virtual channels SpaceFibre operates at 10 times the data rate of SpaceWire and can run over fibre optic (up to 100 m) or copper media (up to 10 m).

2.1.2 SpaceWire

This section describes in detail the main interconnection network onboard spacecraft, SpaceWire, and present related protocols, technologies and devices that are used extensively in this thesis. A list of SpaceWire devices used in the experiments described in this thesis is provided in Appendix B.

2.1.2.1 Overview

SpaceWire is an interconnection network composed by nodes, links and routers that supports arbitrary network topologies and different routing policies using credit-based flow control.

2.1.2.1.1 Nodes, Links and Routers

In SpaceWire standard, nodes are defined as terminals, the end points of the network [ECSS 2008]. When a node wishes to communicate data to another node, it sends a packet containing the data over the network. The network delivers the packet to the destination node in a single hop or in multiple hops using SpaceWire routers. The size of the packets is entirely determined by the applications and the storage space available in the nodes.

A network is built out of switching elements interconnected by physical channels, also called links. A switching element, routing switch, called SpaceWire router, has a number of input and output ports. Its main function is to forward data by establishing non-conflicting connections between input and output ports. Therefore, packets are delivered between nodes by making several hops across several shared links and routers.

2.1.2.1.2 Network Topology

With SpaceWire it is possible to build any arbitrary topology so network designers can try to match the topology of the network to the data communication of the problem at hand [Parkes 2008b]. However it should be realised that a special purpose network is not always the best solution as the traffic load on such networks is in some cases poorly balanced. Sometimes it is better to use a good general purpose network than to design a network with a topology matched to the problem. In any case, a good topology should exploit the characteristics of the network to meet the bandwidth and latency requirements of the application at minimum cost.

2.1.2.1.3 Routing Policies

The sequence of hops across the network define the network path. SpaceWire assumes a single destination per packet, i.e. unicast communication, and it can implement different routing policies (source, distributed and adaptive) based on the user requirements.

The routing is classified as source (centralised) or distributed depending on where the routing decisions are taken. With source routing, the exact path taken by a packet is known before the packet is injected in the network. The routing decision is taken either by the source node or by a routing function that is central for the network. The packets sent by the source node have a packet header containing not only a destination address, but also a description of the path, the destination path addresses. Each router on the path reads the packet header in order to determine in which direction to forward the packet. With distributed routing the packet header contains the destination address but no description of the paths to take. Each router decides from the destination address in which direction to forward the packet, thus the routing decision is distributed among the routers in the network.

In most network topologies there is more than one possible path between any pair of nodes. The procedure of selecting a path is governed by the routing algorithm which is classified as deterministic, oblivious or adaptive. Deterministic routing always chooses the same path between a source-destination pair even if there are multiple possible paths. Oblivious routing chooses a path independent of, i.e. oblivious to, the network state, however, the choice is not necessarily deterministic, e.g. alternates between two output ports of a router. Adaptive routing algorithms choose a route taking the current network state into consideration. They adapt their decision to the state of the network as the usual goal is to balance the network traffic load, to increase the network throughput and to reduce the packet latency. Figure 2-6 summarises this classification.

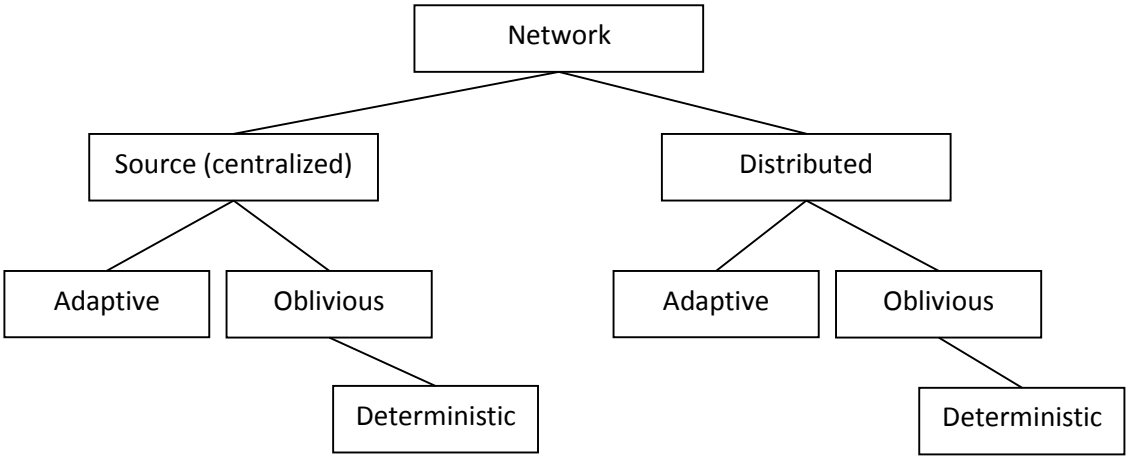
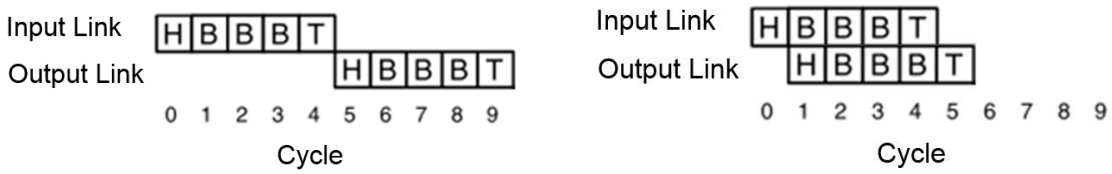


Figure 2-6: Classification of routing policies

2.1.2.1.4 Flow Control

Flow control manages the allocation of resources to packets as they progress along their path or route. Packets are divided into smaller fixed-size data units called flow control digits, or flits. A flit is the smallest unit of information recognised by the flow control. In SpaceWire a flit contains one byte of data.

There is a choice in the granularity at which flow control mechanisms allocate each of these resources. With traditional store-and-forward flow control, each node along a route waits until a packet has been completely received (stored) and then forwards the packet to the next node. Instead, SpaceWire uses Wormhole Switching or Wormhole flow control [McKinley 1993] that works at flit level. SpaceWire overcomes the latency penalty of store-and-forward flow control by forwarding a packet as soon as the header is received without waiting for the entire packet to be received. Figure 2-7 illustrates a time-space diagram of these techniques showing the header (H), body (B) and tail (T) of the packets.



a) Store-and-forward b) Wormhole Switching

Figure 2-7: Difference between Store-and-forward and Wormhole Switching

SpaceWire minimises the required buffer space by allocating buffers in units of flits instead of in units of packets. More precisely, a SpaceWire codec sends one Flow Control Token (FCT character) each time eight flits or data bytes are received. Besides the advantage of reduced buffers, this technique decouples the packet length from the buffer size. However, in a wormhole network a router can buffer only part of the packet so the body of the blocked packet spreads over multiple routers along the path occupying one link per router. Thus, a blocked packet can reduce the throughput and increase the latency for other packets that request the same links.

2.1.2.1.5 Packet Format

The SpaceWire standard defines the following basic packet structure:

<destination address> <cargo> <end of packet>

Where:

- The destination address consists of a list of one or more bytes, called destination identifiers: <destination address> = <id 0> <id 1> ... <id N-1>
- The cargo contains zero or more bytes
- The end of packet is either an EOP, indicating a normal termination of a packet, or an EEP, indicating a packet in which an error has occurred.

2.1.2.2 RMAP Protocol

SpaceWire standard defines only a data link protocol layer. The SpaceWire Transfer Protocol Packet Structure [ECSS 2010] specifies a way to encapsulate upper level protocols in the SpaceWire packet structure using a protocol identifier.

The most important upper level protocol is the Remote Memory Access Protocol (RMAP), used to write to and read from memory, registers, FIFO memory, mailboxes, etc, defined in a destination node on a SpaceWire network [Parkes 2006]. RMAP is both a transport and an application layer protocol and is encapsulated in SpaceWire packet format using a specific protocol identifier.

All read and write operations defined in the RMAP protocol are posted operations i.e. the source does not wait for an acknowledgement or reply to be received. This means that many read and write operations can be outstanding at any time, but there is no timeout mechanism implemented in RMAP for missing acknowledgements or replies. If an

acknowledgement or reply timeout mechanism is required it must be implemented in the source user application. Figure 2-8 shows the write command packet format. A complete description of all packet types is available in the RMAP standard [ECSS 2010b].

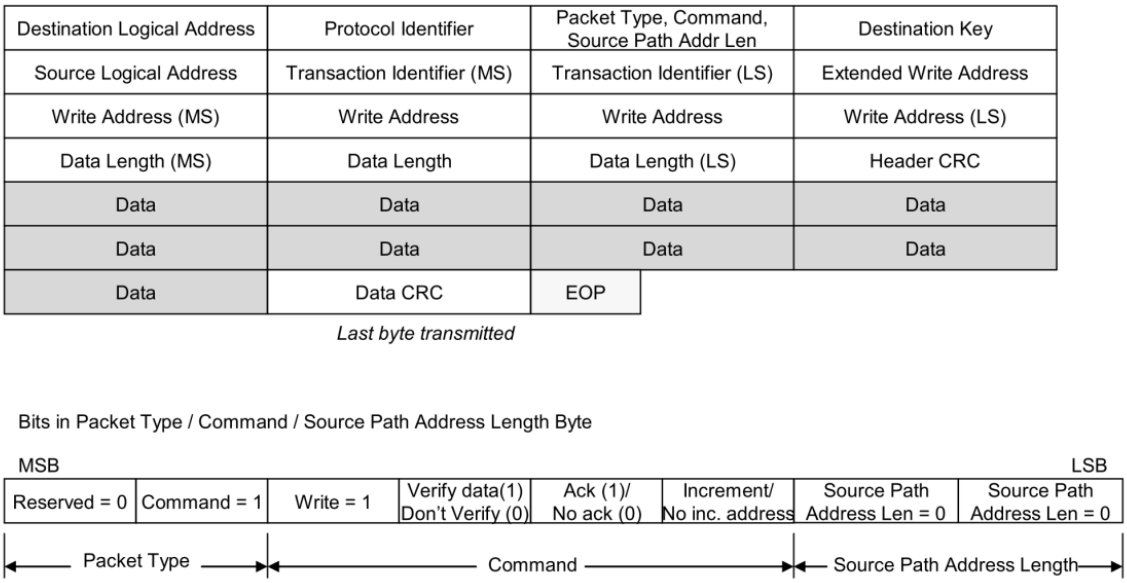


Figure 2-8: Write Command packet Format

2.1.3 SpaceFibre

SpaceFibre is a new networking technology being developed for onboard spacecraft that targets much higher data rates than SpaceWire. SpaceFibre achieves gigabit data rates using 8b/10b encoding [Widmer 1983] which is also used by other protocols such as USB 3.0 [USB 2008] and PCI Express [Budruk 2003].

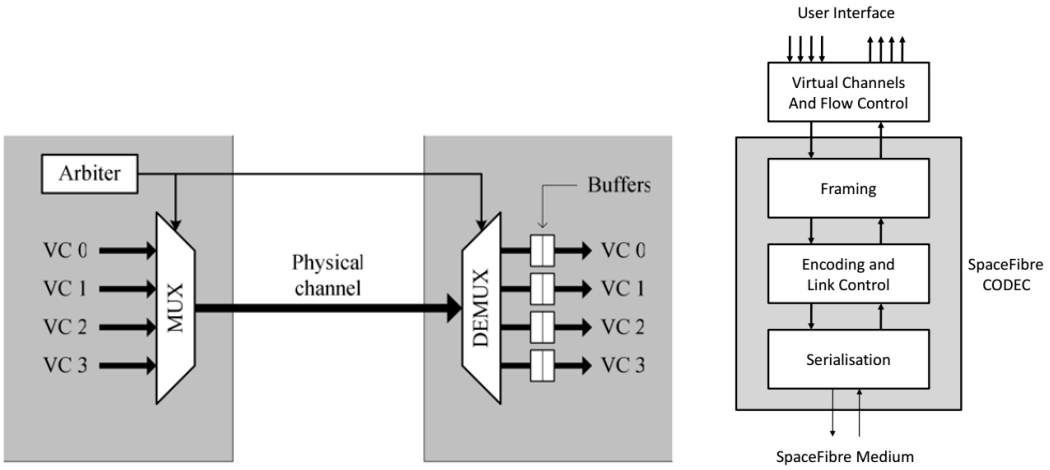
8b/10b is a line code that maps 8-bit symbols to 10-bit symbols to achieve DC-balance, i.e., difference between the count of 1s and 0s in a string of symbols is no more than 2, and yet provide enough bit transitions to allow reasonable clock recovery using a Phase-Locked Loop (PLL). The 8b/10b code also provides 12 control symbols that can be used for link-layer protocol operation.

One key feature of SpaceFibre is that it sends and receives SpaceWire packets using the same format specified in section 0. This allows full compatibility with SpaceWire user applications software. The SpaceFibre codec takes a SpaceWire data stream and chops it into frames containing multiple SpaceWire packet bytes delimited by control symbols. The use of framing allows to implement the concept of Virtual Channels (VC).

2.1.3.1 Virtual Channels

Using virtual channels, several logically independent channels can share the same physical channel, improving the performance of a wormhole network [Dally 1992]. In SpaceFibre, each virtual channel can send and receive SpaceWire packets independently. They allow SpaceFibre to improve network throughput compared to SpaceWire wormhole switching, while still keeping the required buffer space small and the packet length independent of the buffer size.

Figure 2-9 shows SpaceFibre virtual channels in the context of a SpaceFibre codec. The Framing block converts SpaceWire packets from a virtual channel user-interface into frames. Frames belonging to different virtual channels are interleaved in the SpaceFibre link depending on their status and the arbiter operation.



a) SpaceFibre virtual channel arbitration b) SpaceFibre basic block diagram

Figure 2-9: SpaceFibre Virtual Channels

2.2 Quality of Service

Quality of Service (QoS) can be defined as a set of services provided by the network to the demanding network client or application, related with the performance metrics of the network. The services are defined by one or more parameters, which can be low latency, high throughput, low power, bounds on jitter, etc. The services are negotiated which implies balancing the service demands with the services available from the network [Gozdecki 2003].

Services or traffic classes fall into two broad categories: guaranteed and best effort services. Guaranteed service classes are guaranteed a certain level of performance on the services they provide as long as the traffic they inject complies with a set of restrictions. The restrictions usually set an upper bound on some network metrics on the corresponding traffic class offered to the network. The drawback of using guaranteed services is that they usually require resource reservations for worst-case scenarios. In contrast, best effort services do not give guarantees about the services provided. They are designed for average case scenarios instead of worst-case scenarios. Services classes can also be derived from typical application requirements using service levels with different priorities [Bolotin 2004].

Some typical services that may be provided by the services classes are [Avasare 2000]: *data integrity*, meaning that data is delivered uncorrupted, *Lossless data delivery*, which means no data is dropped in the interconnect, *in-order data delivery*, which specifies that the order in which data is delivered is the same order in which it has been sent, and *throughput* and *latency* services that offer time-related bounds. Other complementary services such as *end-to-end flow control* and access regulation to modules in high demand by other units [Walter 2007], involves the management of resources at endpoint, in

particular buffer memory, and for some interconnection networks they can have a large impact on the performance of other services.

2.2.1 Real-Time Communication

With real-time communication the user can either predict the maximum delivery delay for a given message or guarantee the delivery of messages within their timing requirements. Two categories are defined, hard real-time and soft real-time.

In a hard real-time or immediate real-time system, the completion of an operation after its deadline is considered useless - ultimately, this may lead to a critical failure of the complete system. A soft real-time system on the other hand will tolerate such lateness, and may respond with decreased service quality (e.g., dropping frames while displaying a video).

To achieve the Quality of Service (QoS) required for real time communication, researchers have proposed various resource reservation and priority-based scheduling mechanisms i.e., to provide guarantees in latency and bandwidth [Milojevic 2008]. Different mechanisms such as Time-Division Multiplexing (TDM), Virtual Channels and Virtual Circuits are detailed in chapter 4.

The network cannot always accept all the traffic generated by the data sources leading to an increase in the latency. The traffic will have to wait for a necessary resource to become free. Therefore, increasing the throughput typically increases the amount of contention, and it can lead to an exponentially growing latency even when the offered traffic is not maximised Figure 2-10 shows how the saturation point is lower than the network capacity.

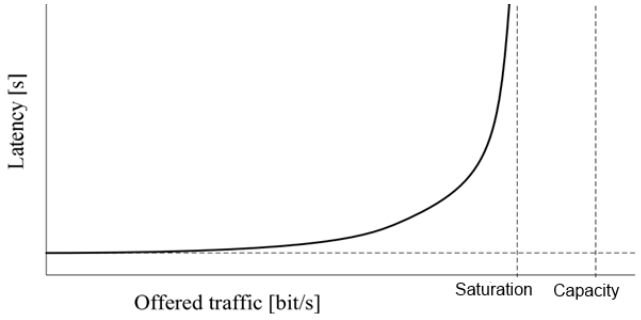


Figure 2-10: Typical dependency between offered traffic and latency

2.2.2 Reliability and Fault Tolerance

Many applications of interconnection networks require high reliability and availability. Thus, they require reliable networks that employ error control techniques to continue operation without interruption, and possibly without packet loss, despite the transient or permanent failure of a component.

2.2.2.1 Failure Modes

Failures that may occur in a system can be classified as failure modes. Failure modes do not include packets dropped by a flow control method although they will also affect services related with reliability, such as data integrity and lossless data delivery. Noise on a physical channel, corrosion on a connector, alpha-particle strikes, and software failures are examples of failure modes. In order to hide the unnecessary complexity of failure modes, simple fault models are used with a corresponding unit of measurement.

Transient failures are usually modelled with a Bit-Error Rate (BER) or Soft-Error Rate (SER) and the inverse of these rates is the time between errors. A soft error is one which has no long term damaging effects. On the other hand, hard errors typically cause lasting, non-recoverable damage to the circuitry which cannot be cleared through a system reset or power cycle. Once a hard error occurs, it is permanent. Permanent errors are usually described in terms of their Mean-Time Between Failures (MTBF). They are modelled

with a stuck-at-fault model in which it is assumed that some logical node is stuck at logic one or zero. Other permanent failures are modelled as fail-stop faults in which it is assumed that some component (i.e. link or router) stops functioning and informs adjacent modules that it is out of service. Often, systems are designed to reduce stuck-at faults, or even excessively frequent transient faults, to a fail-stop fault. Components monitor their own execution and shut themselves down when they detect an error. This avoids the possibility that they became a Byzantine fault, where the systems continue to operate, but in a malicious manner, purposely violating protocols that can cause adjacent modules to fail [Dally 2003].

In space applications, Single-Event Effects (SEE) due to the radiation environment are a main concern. The two most important are Single-Event Upset (SEU) and Single-Event Latchup (SEL) [Kamik 2004]. Table 2-1 shows possible failure modes and fault models, where the Medium Time Between Failures (MTBF) is given in units of 10^9 hours (FITs).

Table 2-1: Example of failure modes and fault models

Failure Mode	Fault Model	Typical Value	Units
Gaussian noise on a channel	Transient bit error	10^{-20}	BER (errors/bit)
Alpha-particles strikes on memory	Soft error	10^{-9}	SER (s^{-1})
Alpha-strikes on logic	Transient bit error	10^{-10}	BER (s^{-1})
Connector corrosion open	Stuck-at fault	10^{10}	MTBF (FITs)
Power supply failure	Fail-stop	10^4	MTBF (FITs)
Software failure	Fail-stop or byzantine	10^4	MTBF (FITs)
Single Event Upset (SEU)	Soft error	10^{-8}	MTBF (FITs)
Single Event Latchup (SEL)	Hard error	-	-

2.2.2.2 Error Control

All error control involves three basic steps: detection, containment, and recovery. Once a fault or error is detected it must be contained to prevent its propagation so the system can recover and resume normal operation. The error control can be done simultaneously at link and router level, network level or at end points.

Error detection at the link level is performed by encoding redundant information on the link, using an error control code (ECC). It can be a simple parity, sufficient to detect any single bit error, or a cyclic-redundancy check (CRC) of sufficient length that the probability of a multi-bit error going undetected becomes vanishingly small [Blahut 1994]. The error check can be made at different levels of granularity. Checks over large-sized data units are more efficient but delay detection until the entire data unit is received, thus making containment difficult. This is solved by additional, separated protected, critical control information. Besides, it is a common practice to send idle flits or characters when the link is idle in order to continuously test the link. Containment at link level can be done by masking or notifying the error. Masking contains and recovers from the error but it is more complex and usually involves the retransmission.

Router errors can be detected by duplicating the router logic and comparing an exclusive-OR of representative signals on a cycle-by-cycle basis or by doing consistency checks. It is also useful to divide the router into fault containment regions that can be independently shut down. The failed modules can be replaced while the system is operating (hot swapping) although the granularity of replacement (field replaceable unit or FRU) may be larger than the containment unit. Replacement using overprovision is attractive if the system is not accessible (e.g. in orbit). At the network level, link and router failures are modelled as fail-stop and links and routers must route packets around

these failed components. The out-of-service links are flagged as not available and packets are routed using the remaining in-service links [Dally 2003].

If a packet is dropped during the containment of a link or router failure, it may be recovered by using end-to-end packet retransmission [Saltzer 1984]. As the transmitter transmits each packet, it retains a copy in a transmit packet buffer until correct reception is acknowledged. If a timeout expires before an acknowledgment is received, or if a negative acknowledgment is received the packet is retransmitted. The acknowledgment itself must be checked for errors and retransmitted if it is incorrect. Retransmission of only the faulty packet requires reordering so it is easier to simply roll-back transmission to the faulty packet and retransmit all packets sent before the failure occurred. Duplicate packets can be received when an acknowledgement is lost or delayed and they should be dropped by giving each packet a serial number. Out of order packets may be received if adaptive routing or path diversity is used, in which case forcing a retry could be easier than perform packet reordering.

Finally, the clients of the interconnection network, such as processing nodes, network line cards, or I/O devices, are still subject to failure.

2.2.3 End-to-end Resource Reservation

Interconnection networks deals with transfer of data from a source to a destination so the most important resource to be reserved is the buffer space at the destination. This is done using end-to-end flow control.

End-to-end flow control is used to regulate the rate of communication between two terminals through a switched network. In a switch-based system it is possible to transmit packets into the switch faster than they can be delivered to the destination. End-to-end flow control ensures that there is always space at the destination buffer before sending a packet. Therefore packet blocking cannot occur when the sender sends data faster than the receiver can process. This is critical in order to avoid very high network congestion with wormhole switching. Besides, there is little sense in sending a packet into the network if it cannot be accepted by the destination.

Another advantage is the avoidance of some deadlock situations. When implementing message segmentation and multiple channels, it is possible that the receiver application is unavailable for other channels until it has completely received the current message. End-to-end flow control ensures that an application is not deadlocked because of this limitation.

2.3 Quality of Service for Spacecraft Avionics

The data link protocols typically used for Spacecraft avionics for onboard communication, presented in section 2.1.1.1, have very different Quality of Services capabilities and they use different approaches to achieve a certain level of reliability and timeliness.

MIL-SDT-1553 and CAN bus are low-speed serial buses targeting avionics platform related functions. MIL-SDT-1553 is the most robust, using scheduling, retry mechanisms and link redundancy. CAN bus is instead event based, with strong error detection features, using signal level priorities to improve timeliness for high priority channels.

On the other hand, payload avionics are based on high-speed networks based on SpaceWire, which offer error-detection capabilities using parity-bit checks. However, SpaceWire error recovery procedures can lead to user data loss, and its high speed provides good average packet latencies but does not gives guarantees for high priority data flows.

The trend in future spacecraft avionics is to integrate payload avionics with some platform functions or at least use the payload avionics network for command and control of the payload instruments. Several studies have been done to find suitable alternatives that can meet this requirement, specifically commercial ground technologies such as SAFEbus, Fibre Channel, and variations of Ethernet [Hegarty 2005].

These studies show that communication architectures specifically targeted for hard real-time control generally do not provide the data throughput necessary for transporting and managing the large amounts of data that are expected for next generation payload avionics. On the other hand, communications architectures for high-speed, large-volume

data transfer are generally not designed to provide the guaranteed low latency and high reliability required for safety critical, hard real-time control systems [Gwaltney 2006].

Unfortunately, commercial ground technologies have not been designed to handle the space harsh environment, the limited hardware and software resources available onboard spacecraft, the robustness required to achieve total autonomy operation and the long term support and availability required by some space missions that can last decades.

Therefore, an interesting alternative is to consider the current high-speed link-layer networks based on SpaceWire and add a Quality of Service layer that can gives the reliability and timeliness guarantees required for command and control operations.

2.3.1 SpaceWire Transport Protocols

The following two protocols were designed to provide Quality of Service to SpaceWire networks at the transport layer.

2.3.1.1 SpaceWire GRDDP

The GOES-R Reliable Data Delivery Protocol (GRDDP) was developed in the frame of the GOES-R program with the objective to implement reliable data delivery to SpaceWire for the GOES-R spacecraft and instruments [GSFC 2005].

The protocol supports multiple unidirectional channels called Transport Channels between the same sending and receiving node. All Transport Channels have the same priority but they can contain urgent messages that are sent before other messages of other channels. Errors are detected with a Cyclic Redundancy Check (CRC) or when an acknowledgment packet is not received after a timer timeout elapses. Data packets are retransmitted when an error is detected using the original sequence packet number. Figure 2-11 shows the packet format.

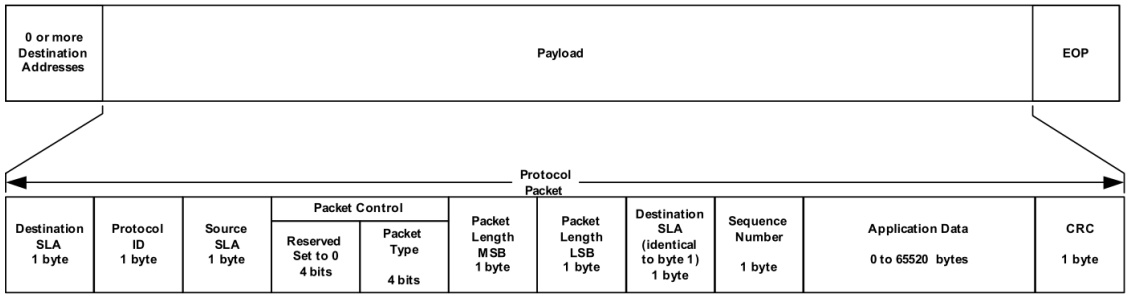


Figure 2-11: Packet format used by the GRDDP protocol

The GRDDP protocol was later evaluated by the Joint Architecture Standard (JAS) program at Sandia National Laboratories suggested a number of modifications to the original specification to meet its program specific requirements [Gardner 2011]. The main modification was additional capability of segmenting user messages that exceeds the MTU size of the protocol.

2.3.1.2 SpaceWire-RT

SpaceWire-RT [Parkes 2008c] is a transport layer protocol intended to provide a Quality of Service (QoS) layer for SpaceWire. SpaceWire-RT stands for SpaceWire Reliable-Timely or alternatively, SpaceWire Real-Time.

2.3.1.2.1 Overview

SpaceWire-RT is being developed as a connection-orientated protocol that can provide end-to-end error detection and recovery, end-to-end flow control and multiple peer-to-peer independent channels between two SpaceWire nodes or terminals. A channel is a unidirectional connection with the following associated elements: an input and output buffer, a set of unidirectional links in the network, a priority value and a set of timeslots in which the channel is allowed to operate. Timeliness is provided using Time-Division Multiplexing and timeslots.

During a timeslot a link can only be used by one SpaceWire source node. This provides deterministic delivery as there are no conflicting shared resources and therefore no congestion arises. The mechanism is implemented in the SpaceWire nodes so it can be used with SpaceWire wormhole switches. SpaceWire-RT also supports asynchronous systems that do not implement timeslots and therefore do not provide deterministic timeliness.

2.3.1.2.2 QoS Classes

SpaceWire-RT provides four Quality of Service (QoS) classes:

- Best Effort: provides a service which does not ensure delivery (i.e. does not provide any redundancy and does not retry in the event of a failure to deliver) and is not timely (i.e. does not deliver information within specified time constraints). It does not deliver duplicate or out of sequence packets.
- Assured QoS: provides a service which is reliable (i.e. retries in the event of a failure to deliver) but is not timely.
- Reserved QoS: provides a service which does not ensure delivery but is timely (i.e. when a packet is delivered it is delivered on time).
- Guaranteed QoS: provides a service which is both reliable and timely (i.e. it will retry in the event of a failure to deliver and deliver information on time).

2.3.1.2.3 Segmentation and Encapsulation

SpW-RT implements message segmentation to ensure that packets sent over the network have a maximum Service Data Unit (SDU) size of 256 bytes. The size of the user information or message is arbitrary and unknown to SpaceWire-RT. The packet format for the Protocol Data Unit (PDU) containing user data is shown in Figure 2-12.

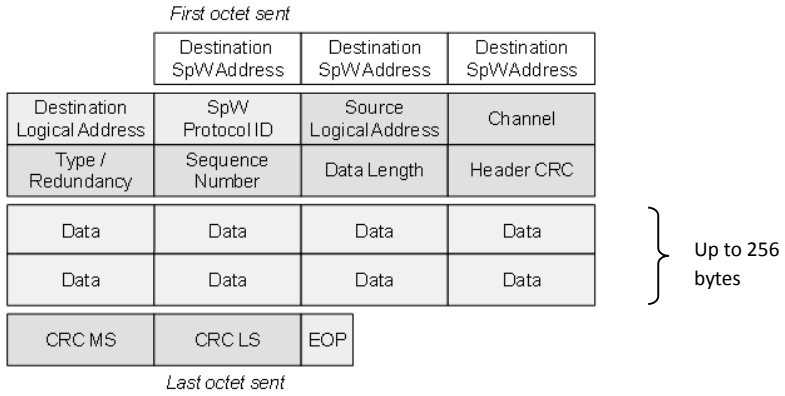


Figure 2-12: Encapsulation of user data in a PDU [Parkes 2008d]

2.3.1.2.4 End-to-end Flow Control

An end-to-end flow control mechanism ensures that there is always space at the destination buffer before a PDU with user data is sent. This prevents the SpaceWire packet containing the PDU being strung out across the SpaceWire network blocking other network traffic when the destination is not ready. Flow control is achieved by the destination channel buffer sending a flow control token (FCT) when it has enough room for another maximum length SDU for that specific channel.

2.3.1.2.5 Error Detection and Recovery

All PDUs contain a CRC for the header and for the data. In order to detect missing and duplicated packets PDUs contains a sequence number and are acknowledged. If an acknowledgement is not received within a certain time-out interval then the PDU is resent. FCTs are acknowledged independently from the PDUs with data.

Furthermore, SpW-RT provides a redundancy mechanism based on the possibility of allowing multiple paths to the same destination. It supports autonomous switching between alternative paths including the possibility to send through multiple paths at the same time.

2.3.1.2.6 Priority

A priority mechanism decides which source channel buffer with data is allowed to send the next PDU to a destination channel buffer that has space for that data. Priority is used for all Quality of Services types.

2.3.1.2.7 Scheduling

Scheduling is provided only for synchronous systems that implement a TDM scheme with timeslots. A schedule table allows multiple channels to be allocated to a specific timeslot but only a single PDU with user data can be sent during this timeslot. Note that Reserved and Guaranteed QoS classes are only available for synchronous systems.

2.4 Quality of Service in other Application Domains

In this section, communication architectures of other application domains are briefly explained, focussing on the elements that are more related to this dissertation. Specifically, the implementation of reliability and redundancy and the differences between event-based and time-triggered communication are described in more detail.

SAFEbus, Time-Triggered Protocol (TTP) and FlexRay, are the primary architectures targeted for safety critical systems.

2.4.1 Aerospace

The aerospace sector has requirements closely related to spacecraft avionics. However, high data rates are usually not required and reliability and safety are the major concern.

SAFEbus is a highly reliable and expensive bus architecture developed by Honeywell and is standardised as ARINC 659 [ARINC 1993]. It is quad-redundant (two self-checking pairs) and each of its four components comprises two data lines and a separate clock line. If the members of a pair disagree, they go offline, ensuring that the failure will be manifest (fail silence). Single-fault hypothesis is assumed. Data transmission is time-triggered and is governed by a message schedule. Synchronised timing of messages delivered is maintained using a global clock. The clocks are synchronised via periodic pulses on the dedicated clock line. It has a limited bus length and a transmission rate of 60Mb/s.

Avionics Full-Duplex Switched Ethernet [ARINC 2005] was developed by Airbus for use in the A380 passenger plane. The AFDX communication protocol adds to the commercial data bus Ethernet (IEEE 802.3) deterministic timing and redundancy management with the goal of providing secure and reliable communications of critical and noncritical data.

2.4.2 Automotive

The automotive industry has addressed the needs of a dependable automotive network for applications like drive-by-wire, brake-by-wire, and power train control. Positioned above CAN in terms of both performance and price, FlexRay is becoming the facto standard against other alternatives like TTP developed at the University of Vienna [Kopetz 1994].

The FlexRay [Flexray 2006] supports communication over single or redundant twisted pairs of copper wire. Despite being a time-triggered protocol it allows asynchronous communication frames to be sent during dynamic segments of a communication cycle which is not supported by TTP. Table 2-2 shows a comparison between FlexRay, CAN and TTP.

Table 2-2: Comparison of FlexRay, CAN and TTP

	FlexRay	**CAN**	**TTP**
Concept	Time/event-triggered	Event-triggered	Time-triggered
Medium access	TDMA plus dynamic segment	Arbitration	TDMA
Topologies	Bus, star, mixed	Bus	Bus, star, mixed
Error containment	Bus guardian	Optional	Bus guardian
Clock synchronization	Distributed, in µs range, offset and rate correction	Optional	Distributed, in µs range, offset correction
Target application	X-by-Wire systems	Operate window, seat control, engine management…	X-by-Wire systems
Speed	Up to 20Mbps	Up to 1Mbps	Up to 25Mbps
Medium access	TDM plus dynamic segment	Arbitration	TDMA
Flexibility	Multiple slots per node, dynamic segment	Flexible bandwidth per each node	Only one slot per node and TDMA cycle
Latency	Bounded by design	Bounded only with probabilistic guarantees	Bounded by design

2.4.3 Supercomputing (SAN, NOWs)

Supercomputers, system-area networks (SANs) and Networks of Workstations (NOWs) are cluster-based parallel computers that rely on high speed interconnection networks. Inter-processor communication with fast message passing requires very low latency while Storage Area Networks demand high bandwidth. In either case, high performance and differentiated services are much more important than reliability and fault tolerance.

Myrinet [VITA 1998] is a lightweight, high-speed local area networking system that implements cut-through switching. Like SpaceWire, it does not use virtual channels and does not provide any mechanism to support QoS. However, it has some fault-tolerance features, including error detection and use of alternate routes to circumvent faults.

InfiniBand is a switch-based point-to-point interconnect architecture where each individual link is based on a four-wire 2.5 Gb/s bidirectional connection [InfiniBand 2000]. Unlike Myrinet, it implements Quality of Service mechanisms through Virtual Lanes (VL) which work like priority-based virtual channels. Other features are Subnet Management Protocol, remote DMA support, multicast and unicast support, reliable transport methods with message queuing, and end-to-end communication flow control.

Fibre Channel [ANSI 1993] is a layered-based interconnect architecture primarily used for storage networking. Fibre Channel supports three basic topologies: point-to-point, arbitrated loop (physical ring), and switched fabric. Fibre Channel includes features which facilitate a “plug and play” discovery process and implements end-to-end flow and reliable transport control using acknowledge frames. Support for differentiated services is provided using a large number of priority levels.

The RapidIO architecture [RapidIO 2008] is a high-performance packet-switched, interconnect technology for interconnecting chips on a circuit board, and also circuit boards to each other using a backplane. It implements hardware-supported segmentation with packets of 256 bytes, with 8-bit or 16-bit addresses, and routed by crossbar switches using point-to-point links. RapidIO supports multicast, four fixed priorities and hardware based error recovery based on acknowledgments and retrials. Besides, it implements end-to-end flow control, memory mapping, mailbox queues, and 8-16 bit interrupt messages.

2.4.4 Network-On-Chip

A Network-On-Chip (NoC) is a lightweight communication network that interconnects the system modules, replacing the traditional on-chip bus of System-on-Chip (SoC). NoC differentiates from off-chip communication in the large amount of wires available for communication and the small area requirement. The following Network-On-Chip architectures have recently been developed and implemented:

The ÆTHEREAL [Dielissen 2003], developed at Philips, is a NoC that provides guaranteed throughput or service (GT) and best-effort (BE) services. GT is provided using a time-division multiplexed circuit switching approach with contention-free routing. It uses wormhole routing with input queuing to route the flits. GT flits are always scheduled for being routed in the next clock cycle, whereas BE flits are scheduled as per a round-robin criterion. The schedule can be reconfigured at run-time. Timeslot tables can be stored in the network interfaces or in routers. In the later case virtual circuits and the associated timeslots can be configured with special BE packet that travels from the source to the destination. The BE packets utilise the resources that are not reserved by virtual

circuits. Credit-based end-to-end flow control has been implemented to make sure that no flit is transmitted unless there is enough space in the destination buffer to accommodate it.

QNOC [Bolotin 2004] aims at providing different levels of Quality of Service for the end users using wormhole packet routing and VCs. These service levels include signalling, real-time, read/write and block transfer, signalling being the top priority and block transfer being the least in the order as listed. The priority-based round-robin scheduling criterion is employed for transmission of flits so strong guarantees are not provided.

SoCWire [Osterloh 2004] is a Network-on-Chip (NoC) approach based on the SpaceWire interface standard to support dynamic reconfigurable System-on-Chip (SoC). It has been developed by IDA, Technical University Braunschweig. SoCWire has been developed to support dynamic partial reconfiguration in future space applications.

Other networks that have been developed with Quality of Service are the the SPIN network [Charlery 2003], the CHAIN network [Bainbridge 2002], MANGO, XPipes networks [Bertozzi 2004], QNOC, SoCBUS [Wiklund 2003]. Table 2-3 summarises the NoC solutions reviewed.

Table 2-3: Summary of reviewed NoC solutions

NoC	Provided services	Flow control	Routing
ÆTHEREAL	BE and GT	Wormhole, TDM	Source routing
Nostrum	BE and GT	TDM	Deflection routing
SPIN	BE	Wormhole	Distributed adaptive
CHAIN	BE, Asynchronous	Wormhole	Source routing
MANGO	BE, GT	Virtual channels	Source routing
XPipes	BE	Wormhole	Static, “street sign”
QNOC	Service levels	Priority based round-robin VCs	X-Y source routing
SoCBUS	BE	Circuit switching with dropping	Distributed, adaptive
SocWire	BE	Wormhole	Source routing

2.4.5 Wireless protocols

There is an additional set of challenges to the implementation of QoS that only exist in wireless or mobile networks [Shakkottai 2003].

The first additional challenge in wireless networks is severe packet loss. Loss in wired networks is typically caused by excessive congestion that causes packets to be dropped at routers in the network. A negligible amount of data is lost due to corruption during transmission on a wire. A wireless link, however, typically suffers much more loss due to data being corrupted during transmission. Another obstacle in wireless QoS involves propagation delay as some wireless networks span distances that are measured in kilometres. Finally, it can be difficult to maintain service guarantees in a network if the nodes involved are mobile. As a result of these additional impediments, QoS schemes used in traditional networks may not be feasible in wireless networks.

The standard 802.11e was the first enhancement to the initial WiFi 802.11 protocol that enabled multiple service levels with the possibility to give service guarantees for network traffic. With 802.11e higher priority traffic always win the contention for the medium and a coordinator can poll the connected nodes giving them an opportunity to transmit following different QoS policies. [Villalón 2005]

The current trend is to implement cross-layer optimizations QoS algorithms where the physical and MAC layer knowledge of the wireless medium is shared with higher layers, in order to provide efficient methods of allocating network resources and applications over the Internet [Liu 2004]. In [Liu 2005], an architecture is proposed which combines QoS reservation and scheduling at the MAC layer with adaptive modulation and coding (AMC) at the physical layer. Even for the new LTE wireless System cross-layering techniques are being developed [Fattah 2009].

2.5 Conclusions

This chapter has presented the state-of-the-art of spacecraft avionics and how Quality of Service is implemented in other application domains.

SpaceWire is currently the preferred high-speed solution for onboard data-handling but the need for gigabit data-rates is driving the development of SpaceFibre which keeps compatibility at packet and network level with SpaceWire. This is an advantage over existing solutions for other application domains which are also not designed specifically for space. Table 2-4 shows the characteristics of high-speed interconnection networks reviewed in this chapter.

Table 2-4: High-speed interconnection networks

	SpaceWire	**Rapid IO**	**Fibre Channel**	**SpaceFibre**
Primary target application	Payload data-handling	Chip-to-chip interconnect	Storage Area Networks	High data-rate payload data-handling with QoS
Speed	Up to 200Mbps	Up to 6.25 Gbauds	Up to 16Gbps	2.5Gbps per lane
Medium access	Event-triggered	Event-triggered with priorities	Event-triggered with priorities	Event-triggered with priorities & Time-triggered
Reliability	Parity bit error detection	Retry mechanism	Retry mechanism	Retry mechanism, redundant lanes.

The need for Quality of Service support for space applications makes it necessary to develop a solution to implement Quality of Service to SpaceWire and SpaceFibre links. SpaceWire GRDDP and SpaceWire-RT are interesting protocols that can be evaluated and improved. In particular, cross-layer techniques may be borrowed from wireless protocols to develop more efficient protocols. Regarding SpaceFibre, the set of QoS requirements defined by current draft specification still needs to be evaluated and implemented.

Chapter 3: Research Questions

This chapter presents the motivation for the research questions and the approach that will be adopted to answer them.

3.1 Quality of Service for SpaceWire

The precise question is: How should a Quality of Service layer be implemented for SpaceWire?

3.1.1 Motivation

State-of-the-art payload data-handling systems are typically based on SpaceWire high data rate interconnection networks. The flexible network topology of SpaceWire can accommodate different use cases and it has enough bandwidth to deal with the current generation of payload instruments and onboard processing units. However, without any additional components, SpaceWire lacks the reliability and timeliness required in some critical command and control operations.

Some commercial ground technologies may meet these requirements but they have not been designed to be used for space applications and their validation and adoption in this field could have a prohibitive cost. A better and simple alternative is to design a Quality of Service layer that works on existing SpaceWire equipment and meets the specific needs of payload data-handling systems.

3.1.2 Approach

The first step will be to review existing QoS solutions used in other protocols and determine which techniques are more suitable to be applied to SpaceWire networks. The

limits imposed by existing devices will be investigated to ensure their compatibility with the proposed techniques. In addition to the theoretical analysis, some simulations or experiments using SpaceWire devices will be carried on to assess the suitability of different solutions.

The second step will be the definition and development of SpaceWire transport protocols that provide the Quality of Service required for the use cases considered, using the techniques previously analysed. The use cases will be inspired by simplified onboard spacecraft architectures in scheduled or asynchronous networks using unidirectional or bidirectional data flows.

For each protocol, a list of requirements will be first presented, followed by a set of design considerations that leads to the protocol specifications. Finally the protocol capabilities and performance will be evaluated using experimental apparatus based on SpaceWire devices in controlled experimental setups.

3.2 Cross–layer Optimizations for SpaceWire

The precise question is: Is it useful to use Cross-layer Quality of Service techniques over SpaceWire?

3.2.1 Motivation

State-of-the-art QoS techniques in commercial ground technologies, specifically in wireless networks, make extensive use of cross-layer optimizations to improve the performance and the QoS metrics obtained.

Cross-layer optimisation removes the strict boundaries between layers in the OSI communication model where data is kept strictly within a given layer. Status information

can flow without restrictions between different layers or they can even be integrated into a unique layer. In the context of SpaceWire, this would allow the transport layer to monitor and control the status of the layer below, the SpaceWire link layer, or implement services of the layer above, the application layer. This should increase the network performance and improve the QoS metrics.

It is expected that cross-layer techniques benefit from a network that it is closed and controlled such as the one implemented in a spacecraft, as the different layers of the network are well known.

3.2.2 Approach

A protocol will be defined, developed and evaluated that unifies the user application layer and the QoS transport layer. The RMAP protocol is an application layer protocol that can be a suitable candidate to give Quality of Service provisions typical of transport layer protocols.

In addition, the possibility of a transport protocol using cross-layer feedback from the SpaceWire link layer will be explored and evaluated with specific protocol specifications. The benefits obtained with each approach in comparison with the results obtained in previous chapters will be highlight using suitable experiment setups.

3.3 Quality of Service for SpaceFibre

The precise question is: How should the new SpaceFibre protocol be designed to provide the required Quality of Service?

3.3.1 Motivation

The SpaceFibre protocol specification draft A [Parkes 2007] defines how gigabit data rates are achieved using 8b/10b encoding and how SpaceWire packets are multiplexed using frames and virtual channels. However, it does not specify how a Quality of Service level can be provided to a determined virtual channel.

There is the opportunity to complete the design of SpaceFibre to provide a new protocol compatible with SpaceWire that provides out-of-the-box Quality of Service at much higher data rates.

3.3.2 Approach

The starting point will be the SpaceFibre specification Draft B [Parkes 2010b] that outlines the provision of Quality of Service using FDIR techniques and a Medium Access Controller that arbitrates between each virtual channel.

The proposed QoS related techniques will be first validated in software simulations as this allows to rapidly check the concepts and provide statistics difficult to gather in a real hardware implementation. Then, working together with STAR-Dundee and members of the University of Dundee a complete prototype will be developed so the protocol mechanisms can be validated in real hardware.

3.4 Conclusions

This chapter has presented the research questions addressed by this thesis:

- How should a Quality of Service layer be implemented for SpaceWire?
- Is it useful to use Cross-Layer Quality of Service techniques over SpaceWire?
- How should the new SpaceFibre protocol be designed to provide the required Quality of Service?

The next chapter four will provide the common grounds to be able to answer each of these questions in chapters five, six and seven.

Chapter 4: Preliminary study of QoS implementation

This chapter presents the main aspects of the Quality of Service paradigm and analyses their application to existing spacecraft avionics based on SpaceWire networks. Note that QoS for SpaceFibre is analysed independently in its own chapter, although some concepts explained here are used.

Two main QoS components, reliability and timeliness are independently analysed. A third component, network discovery and management, is also studied, because it is required to deploy and configure a network with QoS.

For each QoS aspect, the general network concepts presented in the background chapter are briefly reviewed and it is analysed how they can be applied with the highest efficiency to existing SpaceWire networks.

This study also takes into account the possible limitations of existing SpaceWire network components and aims to assess the feasibility of QoS implementations with the development of software prototypes and simulations. The objective is to prepare a solid ground for the actual development of protocols and techniques presented in the following chapters.

4.1 Reliable Communication

4.1.1 Error Handling Mechanisms

Error handling can be classified in three aspects: error avoidance, error detection and error recovery [Dally 2003].

4.1.1.1 Error Avoidance

The most obvious source of errors are link faults produced by the physical channel, but these cannot be avoided. However, there are other sources of errors that can be prevented. Data packets can also be lost due to excessive network congestion. Some routers could be configured to remove packets that are completely or partially stalled in their internal buffers. This could happen because the source is stalled, because the destination is not ready to receive data or because of other stalled packets in a congested network. This last case can be handled by a congestion control mechanism, which aims to decrease the likelihood of packets being removed by routers. This mechanism reduces the injected traffic in the network.

In some implementations, when the destination is not ready, the stalled packet can also be discarded by the destination. It can be because the destination is busy, does not have space to store the data or the resource requested is temporally not available.

An end-to-end (E2E) flow control mechanism can cope with this issue by ensuring that the receiver notifies the sender when it is ready. This implies that there is a flow of information from the receiving side to the sender that is independent from the data flow of the sender. For SpaceWire, it is usual that the receiver has a FIFO where the data received is stored and a user application reads out the data at any time. The receiver must provide to the sender some status information regarding the utilisation of the FIFO.

This status information is not required if the destination is, by design, always ready. This happens for example when the destination is a remote memory unit that uses the SpaceWire RMAP protocol with the incrementing address option. Even if that is not the case and a FIFO is used instead, the benefit of the end-to-end flow control is low when the destination is expected to be ready under nominal operation, i.e. the FIFO is usually

not full. If an error or something unexpected occurs, it could be handled by some error detection and recovery procedure.

In general, the end-to-end flow control mechanism is most useful when the source data rate is variable and the destination buffer cannot cope with bursts of data. For space applications this is not usually the case, as the destination usually has more resources than the source, i.e. the instruments. Still, it can be useful as a notification that the receiving side has read the message sent or in a system that multiplexes multiple destination buffers. Although all this can also be implemented by the application layer, a protocol with an end-to-end flow control mechanism is always more robust and reliable.

Table 4-1 summarises the different use cases.

Table 4-1: End-to-end flow control for space applications

Use case description	Producer	Consumer	Complexity
Source data rate is always less than the consumer rate.	Instrument	Mass Memory Unit	Low
Consumer pulls data from the producer.	Mass Memory Unit	Processing Unit	Medium
Consumer regulates the data rate of the producer.	Processing Unit	Processing Unit	High

Another type of error is produced by the reception of out of order or duplicated user data. Out of order reception occurs when two data segments do not arrive in the same order as they were sent because they followed different paths with different delivery times. Duplicated data occurs when error recovery procedures are in place. These errors are prevented by using sequence numbers to identify each data unit or packet. They are used to indicate any data that is a duplicate of previously received data. For SpaceWire networks with wormhole switching only a few, small, packets can be stored in the routers

and in the links, so the sequence numbers can rollover with much smaller values than it is usually required for other types of networks.

4.1.1.2 Error Detection

When a link error occurs and data is corrupted, it is usually detected at the receiver side using Cyclic-Redundancy Checks (CRCs) or other form of checksum. The receiving side periodically sends positive or negative acknowledgements to the sender side containing data or packet sequence numbers. The sender side can then detect when an error has occurred and proceed with an error recovery scheme. This requires to have a copy of the packets that have not been acknowledged, i.e. the delivery to the destination has not been confirmed. This is done using a sliding window implemented in the sending node buffer. Figure 4-1 shows how the sliding window is updated when one packet is acknowledged.

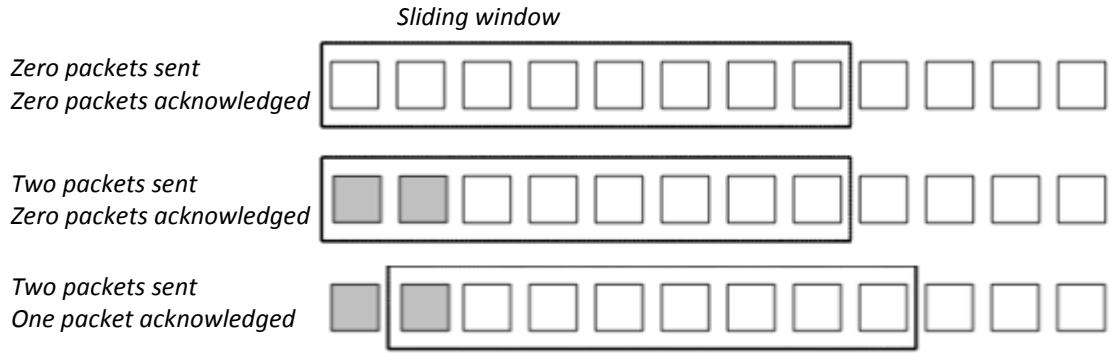


Figure 4-1: Update of the sliding window when one packet is acknowledged

The size of the sending window is the maximum amount of outstanding data, i.e. data sent that have not been acknowledged yet, allowed at any given time. When the sending window is full, the sending node stops sending more data, starts a timeout timer and waits to receive an acknowledgement. If the timeout timer expires without receiving an acknowledgement, it is assumed that all data in the sending window has not reached the destination correctly.

The maximum size of the sending window is limited by the range of data packet sequence numbers. For example, if a single bit for the sequence number is used, the system does not send a new packet before it has got the acknowledgement of the previous packet sent. This approach, send and wait, is a quite effective solution if the reply is expected to be received shortly after the packet is sent and this waiting time is already considered in a scheduled network with real time characteristics. For larger sending windows, the number of possible sequence numbers should be a multiple of the sending window size. This reduces the chances of delayed acknowledgement being considered to be the acknowledgement of a newer packet due to sequence number rollover.

A negative acknowledgement usually reduces the error detection time, as it removes the need to wait for the timeout of positive acknowledgments, when a corrupted data packet has reached the destination. It does not reduce the detection time if the packet is lost before reaching the destination, if the header is corrupted, or if the negative acknowledge is lost. It can also further reduce the recovery time as it allows to identify the packet lost or the error type. However, the extra complexity is usually not worthwhile for a transport protocol over a network for space applications.

It can be noted that the use of negative acknowledgements makes it possible to avoid the use of timeout timers that need to be set depending on the worst case network packet latency. However, this requires the sender to periodically sent data in order to deal with loss of acknowledgement packets, which is a waste of network bandwidth. Therefore, this approach is only used for link-layer protocols and not by transport layer protocols.

For wormhole switching networks, it is common that a data packet is larger than the network buffering. Chunks of a packet are stored across all router buffers in the network, so the receiver starts receiving the packet while the sending side is still transmitting it.

Therefore, if the acknowledgment is sent just after the packet is received and there is no congestion in the network, it should arrive before the second data packet has finished being sent. Figure 4-2 shows, above, a packet being sent by Node A, using all buffers of the routers, and below, the receiving side B generating an acknowledgement using a small packet that fits in only one of the router buffers. Note that the squares with patterns indicate what the router buffers contains.

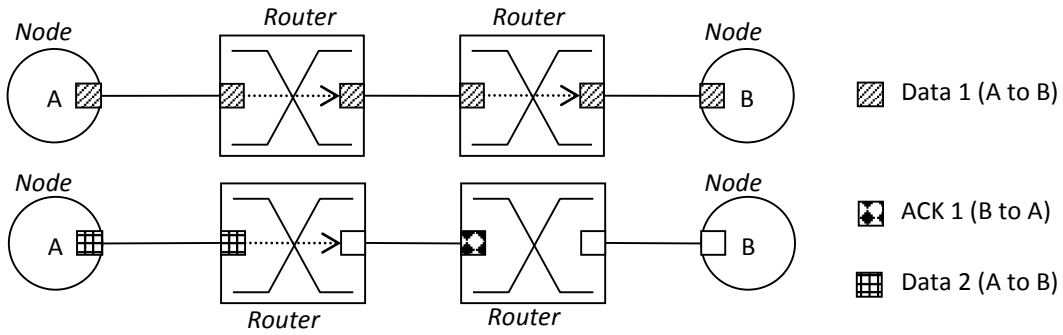


Figure 4-2: Data and acknowledgment packets within the routers buffers.

In this scenario there are never more than two outstanding data packets under nominal conditions. This is a major difference with respect to store and forward networks. For wormhole switching, complex packet reordering and selective retry should not be implemented. The window size is small and the cost of resending all packets stored is not high.

4.1.1.3 Error Recovery

The main mechanisms to recover from transient and permanent network errors are forward error correction techniques, retry mechanisms and the use of alternative paths to the destination. For space applications, the mechanisms implemented should handle any type of networks error and should cope at least with the single fault hypothesis. If it cannot recover from the error, the system should go silent or into another fail-safe state, to avoid

error propagation. Then it should pass a notification to the user or upper layer protocols, where it should assume that the error is caused by a malfunctioning terminal or by multiple network errors (link or router failures).

Avoiding error propagation is very important to ensure that multiple consecutive errors are not due to network congestion but are caused by a permanent link error in the path to the destination. Figure 4-3 shows a congestion error due to a problem in another path. The data flow from the source *Sa* to the destination *Da* becomes blocked due to the other flow being blocked in the link with an error. The router *R1* can decide after some time to remove the blocked packet.

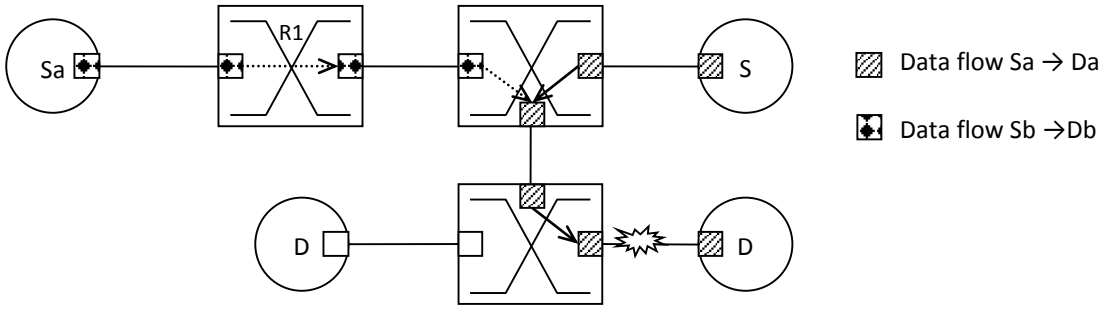


Figure 4-3: Congestion error due to other packet flows.

These errors due to network congestion can be recovered using a retry mechanism. A waiting period should be added after an error is detected to reduce the likelihood of more congestion errors. This procedure is followed by most commercial transport protocols, like the TCP/IP stack, that has the following useful feature: the protocol behaviour is transparent to the upper layer protocols. Protocols for space application should have similar goals, but should be much simpler to implement, taking advantage of the constrained environment. With this philosophy, forward error correction techniques, which are expensive to implement, should not be used. They deal with some cases of data

corruption and transient link errors that are solved more robustly using the retry mechanism.

To recover from permanent link errors, data should be resent using alternative paths. SpaceWire has a interesting feature called Group Adaptive Routing (GAR) that allows a routing table to specify a group of multiple destination ports for a single destination logical address. The packet will be sent to the first link of the group that is available. Therefore, if a link is busy or it is disconnected, another link will be used. This technique is useful for increasing the total bandwidth but should not replace an explicit redundant path scheme as it does not cover router failures or persistent intermittent link failures.

4.1.2 End-to-end Channel Concept

In this thesis, a channel definition will be used that it is similar to the concept of an opened connection in the TCP/IP architecture. A channel encapsulates all the parameters of an end-to-end connection between two entities located in two different terminals of the network. These parameters include, for example, the path used between these two entities, the QoS level, and protocol status values such as data sequence numbers.

4.1.2.1 Unidirectional or Bidirectional Channels

Channels can be unidirectional or bidirectional. In bidirectional channels each connection side is receiving and sending data. With unidirectional channels each side is either receiving or sending. Bidirectional channels can be built with two unidirectional channels but a specific bidirectional channel can take advantage of the piggybacking technique and the sharing of common configuration parameters such as network paths. Bidirectional channels are especially efficient if the user application performs command and response

operations using the RMAP protocol over SpaceWire full duplex links as they require fewer packets to be sent for each RMAP command.

4.1.2.2 Medium Access Control

When a terminal has multiple channels that are sending data, the question arises of how to arbitrate the access to the physical link. It can be based on a priority list, where the highest priority channel sends its data first, a fair arbitration mechanism like round-robin, or a bandwidth allocation scheme. In most SpaceWire units, a higher priority packet must wait for a lower priority packet to finish being transmitted before it gains access to the link.

4.1.2.3 Channel Setup

Most robust transport protocols must set up some channel parameters before data can be received. They are called connection-oriented protocols. The sender requests the receiver to open a connection that the receiver must acknowledge. It can require two or three steps, in which channel parameters, such as sequence numbers are initialised. The problem is that this process is an overhead when the connection duration is short and may involve the use of complex state machines. For space protocols, the amount of configuration parameters may be small. Then, this setup phase could be avoided and the initialisation information could be included in the header of data packets in a connection-less protocol.

4.1.3 Data Encapsulation and Packetization

Transport protocols implement reliability mechanisms by encapsulating protocol status parameters within the packet structure. This allows the communication to be independent of lower layer protocols, more precisely the link layer. The unit of information is the packet.

The status of the channel can be encapsulated within the header of a data packet or in a separate control packet. Typically, parameters such as the sequence number are combined with the data and are sent together. On the other hand, end-to-end flow control information is generated independently, and can be sent separately. In any case, control and status information must be as reliable as the user data, so independent acknowledgements have to be generated with timeout mechanisms that are usually implemented at the sender side.

4.1.4 Experiment: SpaceWire Reliable Protocol

A software prototype was developed as a proof of concept of how to provide a reliability layer for SpaceWire. It is based on the initial specification of the SpaceWire-RT protocol presented in the background chapter [Parkes 2008c] which implements multiple concepts of this subchapter. It provides end-to-end error detection and recovery, end-to-end flow control and multiple peer-to-peer independent unidirectional channels. The objective was to identify possible improvements to support the development of the new protocols presented in the next chapter.

4.1.4.1 SpaceWire-RT

The SpaceWire-RT protocol defines four types of packets to cover the reliable delivery of user data and end-to-end flow control. The sender transmits these packets with the following priority order:

1. Acknowledgment packet (ACK): Contains the sequence number of the last Data packet received.
2. Acknowledgment of flow control packet (BACK): Contains a sequence number of the last BFCT received.

3. Flow control packet (BFCT): It is sent when the destination buffer has space for another data segment. It contains a sequence which increments each time another BFCT is sent for a specific channel.
4. Data Packet (DP or PDU): contains a segment of a user message in a Service Data Unit (SDU) and a sequence number that is increased for each data packet sent. The maximum size of a SDU is 256 bytes.

When a DP or BFCT is sent, a timer is started. It is cancelled when the corresponding acknowledge is received with a valid sequence number. If the timeout timer expires, the DP or the BFCT is resent. The DPs and the BFCTs implement a sending window. A BACK or ACK can acknowledge multiple BFCTs or DPs. An example with BFCTs is shown in Figure 4-4 where the BACK with sequence number 10 acknowledges the BFCT with sequence 9.

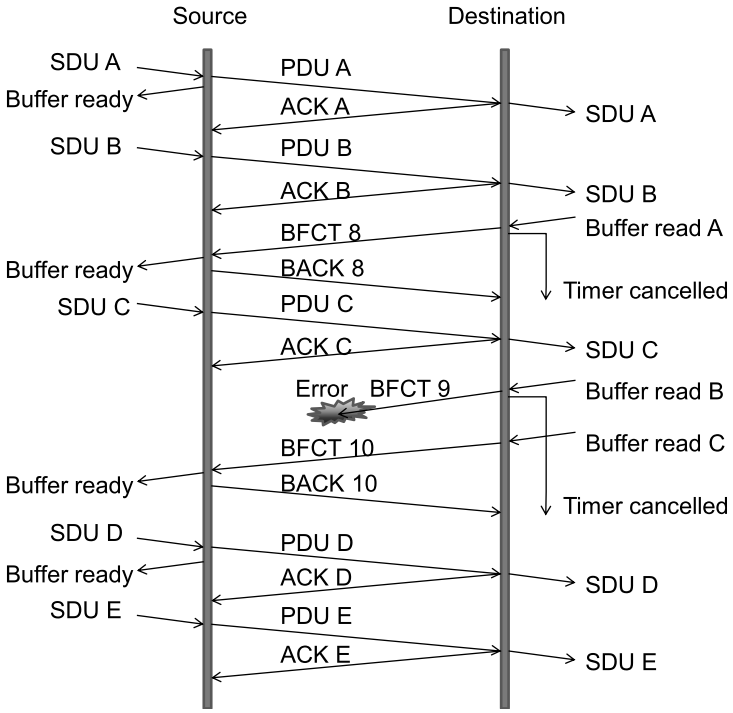


Figure 4-4: SpW-RT initial specification sequence diagram [Parkes 2008d]

4.1.4.2 Implementation

The prototype was written in Python [Rossum 2003], that can be used for fast prototyping and simulations that do not require high performance. The objective was to prove the logical consistency and the reliability capabilities of the protocol, not the performance of the implementation.

In order to support SDUs of different sizes, the system has two counters that keep track of the number of data bytes sent and data bytes acknowledged. The system also stores the sequence numbers sent and received for each packet type. Each sequence number represents 256 bytes, so it is generated or updated when the difference between each pair of byte counters exceeds this threshold. The sequence number is checked using a sending window of three SDUs. If a duplicated DP or BFCT is received, an ACK or BACK with the expected sequence is sent. This supports the possible loss of these acknowledgment packet types.

The segmentation function includes the capability to indicate the start and the end of the original user message. The redundant path function is applied as follows: the BFCTs and the DPs uses another path after a maximum number of errors and retries. A packet field indicates the path to be used by the ACKs and BACKs related with these BFCTs and DPs.

4.1.4.3 Optimization

Two optimizations were designed and applied to the original protocol specification

1. Instead of starting an independent timer for each DP or BFCT sent, a single timer is used for each type of packet. If a timer is active when sending a new DP, the timer is restarted. This reduced the implementation complexity.

2. Instead of forcing the receiver to send a BFCT for each maximum SDU size, read by the receiving application, a single BFCT can notify the read-out of multiple SDUs. The sequence number of the BFCT is not always one more than the previous one sent. Instead, it is increased by the number of SDUs that have been read-out since the last BFCT was sent. This reduced the amount of packets sent leading to an increase of the data throughput measured.

4.1.4.4 Results

The prototype was first tested with a point-to-point loopback connection. A Link Analyser with the capability to inject errors on the link was inserted between each end point. The source and the sink data rate measured followed a Poisson distribution. Received data was checked and no data loss was detected during many hours of operation under continuous link errors when using an unlimited number of retries. This proved that the protocol could recover from the loss of any packet type.

Another test setup was used to verify the implementation of redundant paths. Figure 4-5 shows the test equipment consisting of an experimental apparatus (Node A) connected to a router with five ports and three links. Two different loopback paths are set. The nominal path uses Link 1 and Link 2. The redundant path uses Link 1 and Link 3. A Link Analyser is inserted in Link 3 to monitor its status and inject errors.

After starting the system, Link 2 was disconnected, so the system switched to the redundant path. Then some transient errors were injected in Link 3, which the system managed to recover. Finally, Link 3 was also disconnected and the system notified a non-recoverable network error.

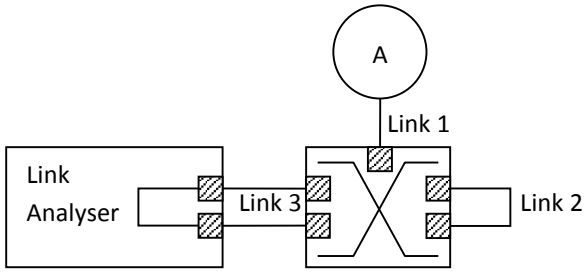


Figure 4-5: Prototype test setup

4.1.4.5 Lessons Learned

The use of different packet types for each function (ACK, BACK, BFCT, DP), although it is conceptually simpler, it is not as efficient as encapsulating multiple functions in a single packet. Each packet sent implies a delay that increases the user data latency and throughput. For example, using piggybacking and bidirectional channels, the acknowledgment and the data can be encapsulated in a single packet. Furthermore, the status of multiple channels can be encapsulated in the same packet if they are from the same source-destination pair. These possibilities will be examined in the next chapter.

4.2 Timely Communication

Providing timely or real-time communication in an event-based network with wormhole switching is a challenging task. Soft real-time, i.e. low latency without hard guarantees, can only be easily achieved without specific support when the network load is very low, typically less than 10% for a grid topology with uniform traffic. In that scenario, wormhole switching provides lower latencies than the store and forward technique. However, the network quickly saturates starting with a uniform traffic load of 30%, with latencies much higher than the ones provided by store and forward, which handles higher loads better [McKinley 1993].

The fact is that real-time is not about the minimum latency measured, which is very low for wormhole switching networks, but about the guaranteed maximum latency. Average latency is also important if the application is non critical, i.e. is not hard real time. The average latency also depends on the load of the network and the average throughput.

This section will introduce some new techniques for network analysis in wormhole switching networks, in particular the computation of latency and throughput metrics. Then, possible solutions will be studied to improve these metrics and provide possible guarantees.

4.2.1 Latency and Throughput

4.2.1.1 Throughput and Network Saturation

When a network is designed, an analysis is performed to adjust the capacity of the network to the expected data rates of the different data flows. A network is saturated when the offered traffic is higher than the accepted traffic or throughput. The analysis of

wormhole switching networks can be tricky when using concepts borrowed from classic store-and-forward networks.

One common mistake occurs when working with a network that has different link speeds. Figure 4-6 shows a SpaceWire network with different link speeds. Nodes *A,B,C* are sending data to node *D* at the data rates specified. In store and forward routers, this network would have a reasonable load. The user data rates are lower than the link speeds and the sum of the input link speeds (25+10+50) is less than the output link speed (100 Mps). However, in a wormhole switching network this configuration will not work, as the sum of link utilisations (%) is higher than 100%. The fact is that while the node *A* is sending data to node *D*, the link speed set to 100Mps is operating as if it was running at 10Mpbs, i.e. it has many SpaceWire Null characters between each SpaceWire data character to give the 10Mbps data rate.

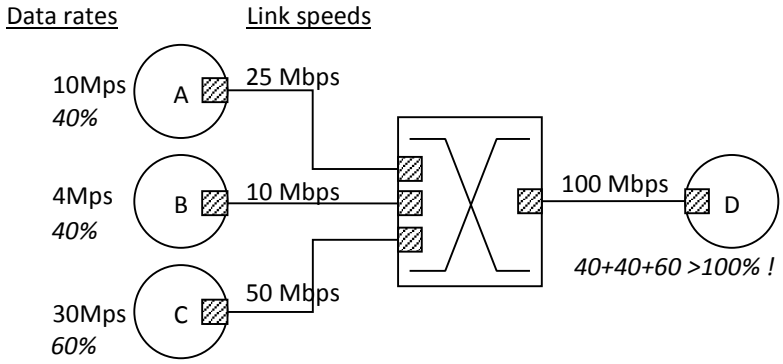


Figure 4-6: Saturated SpaceWire network with different link speeds

Therefore, the following condition must be met to ensure that the network is not saturated when multiple input flows go to the same destination:

$$\sum_{i=1}^{Nsources} \left(\frac{R_i}{S_i}\right) < 1 \qquad (4.1)$$

Where R_i is the user data rate and S_i is the minimum net link speed across the path from the source *i* to the destination. For SpaceWire the net speed is 80% of the raw link speed.

Another typical error, without any relation with the previous one, is to perform an analytical study based on summing up for each link, the data rate of each data flow that will use it, and checking that it is lower than the link capacity. This method fails because it does not take into account the bandwidth wasted due to blocking effects. For example, in Figure 4-7, the flow going from node *A* to node *D* is blocked due to flow going from *C* to *D*, so meanwhile the bandwidth of link *L1* is not being used. Therefore, in this case, for ensuring that the network will not saturate due to congestion in link L1, the capacity of link *L1* has to be higher than the sum of the source data rates of all three flows.

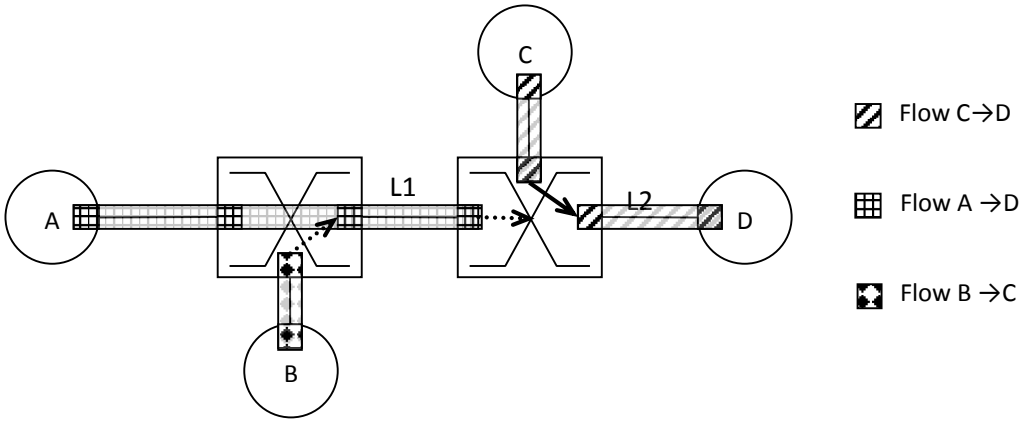


Figure 4-7: Example of link bandwidth wasted due to network congestion

4.2.1.2 Latency Computation

As stated, the computation of the expected latency cannot be done with the same methodology used for store and forward networks. The techniques used to obtain the minimum, the average and the maximum latency need to be revised. Note that in this section the term latency refers to the transmission delay, so the time a packet may be stored in a source node queue, waiting to be sent, is not accounted for.

The minimum latency is the one obtained when there is no other traffic in the network besides the traffic of the flow considered. In the absence of congestion, the minimum latency is determined by equation (4.2), using the capacity of the link (C), the switching latency of the routers (T_{sw}), the number of hops and the packet length of the flow (L). A new hop is considered each time a packet is routed to a new link.

$$D_{min} = \frac{L}{C} + \sum_{hops} T_{sw} \qquad (4.2)$$

The average latency computation must take into account the congestion of the network. Therefore it requires to know the network topology and the traffic injected in the network. Quite accurate values can be obtained with statistics parameters about the traffic injected. The results are computed using a network simulation tool. In section 4.2.8 a simulation experiment is presented.

An upper bound of the maximum latency can be computed analytically only with the topology of the network and the maximum packet size per flow. The actual network traffic does not matter here because the worst case is assumed, in which the network is fully occupied by packets in all links. Also note that as the latency obtained with this method is defined as an upper bound, it is higher than the worst case maximum latency usually measured and it can be higher than the maximum latency that can occur in a specific network. However, it is still useful as it can be used to prove analytically that certain network requirements are met.

The actual computation needs to be performed iteratively using equation (4.3). The latency of one flow depends on the network blockage time encountered at each hop until the destination. This latency per hop is the sum of the worst case latency of flows using other input ports of the router.

$$D\ (F,H) = \sum_{\substack{h \in hops \\ left}} \left\{ T_{sw} + \sum_{\substack{p \in input \\ ports}} MAX[D(f,h)]_{|\forall\ flow\ f\ using\ p\ and\ h+1} \right\} \qquad (4.3)$$

The term $D(F,H)$ is an upper bound for the worst-case latency of a flow F from a particular hop H up to the destination. The computation should start by calculating for each flow the last hop to the destination and continue backwards until the first hop (zero), which provides the total upper bound latency. The value for the last hop is given by the following equation:

$$D\ (F,H_{last}) = \frac{L}{C} \qquad (4.4)$$

Note that in equation (4.3), flows f that use any of the links used by flow F should not be included. Also, note that if the flow considered is processed by the router with higher priority than other flows, then the summation term for each input port should not be included. Finally, if a source can send multiple flows, the arbitration between them should be considered, taking into account the worst case. For example, if priority mechanisms are used, the latency of a high priority packet will depend on the latency of lowest priority packet. For example, the low priority packet could just start being sent when the high priority packet is created.

Figure 4-8 shows an example of applying equation (4.3) to a simple linear topology.

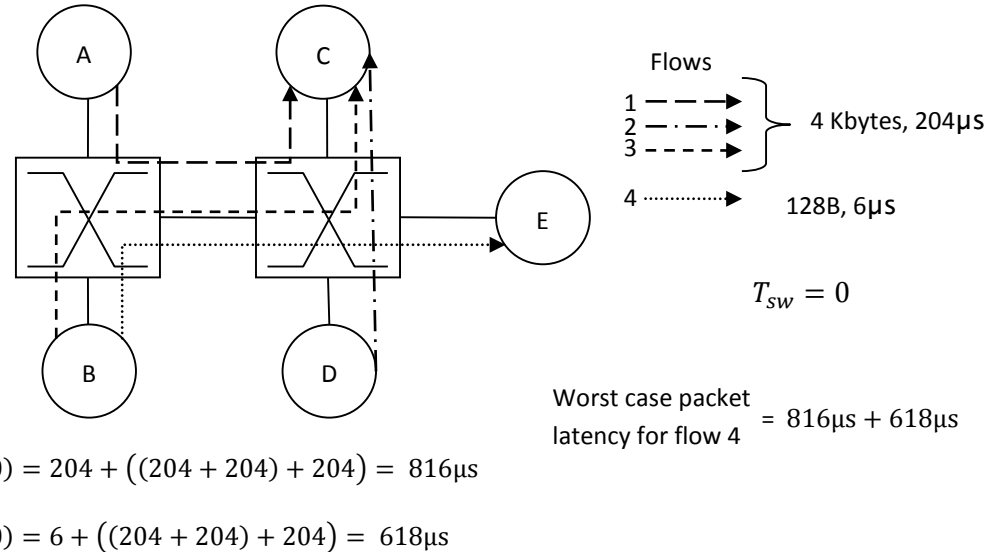


Figure 4-8: Simple example of a worst-case latency computation

As stated, the latency of packets in wormhole switching depends on the distance to the destination. For linear topologies the worst case latency can be computed with the following simple non-recursive expression that shows that the upper bound of the maximum latency increases exponentially with the number of required hops.

$$D = \left[\mathrm{N_p} * \left(\frac{\mathrm{L}}{\mathrm{C}} + T_{switch}\right)\right]^{\mathrm{N_h}} \qquad (4.5)$$

Where $\mathrm{N_p}$ is number of ports and $\mathrm{N_h}$ is number of hops.

4.2.1.3 Guaranteed Throughput

As stated, the throughput of a specific flow depends on the network congestion. It is interesting to see that a minimum throughput can be guaranteed under any network load conditions, including a completely saturated network. It is equal to the packet length divided by the maximum latency, as shown in equation (4.6). A lower bound for the throughput is obtained if an upper bound for the maximum latency is used.

$$Th_{min} = L/D \qquad (4.6)$$

If different flows use different packet lengths, the guaranteed throughput can be increased by either increasing the packet length or reducing the maximum latency. The latency can be reduced by changing the topology or reducing the packet length of other flows.

4.2.2 Segmentation

Equation (4.6) shows that the packet length is a key parameter that determines the latency. It is a parameter that can easily be modified by the sending entity, using segmentation. The packet size depends on the segment size plus some header overhead and it does not depend on the message size.

4.2.2.1 Segment Size

A segmentation layer should be transparent to the user application and therefore, its parameters should be based on the network characteristics. However, the basic segmentation parameter, the length of a segment, has to consider the typical size of the user messages with latency requirements. These messages are usually used for command and control and are smaller than the ones that compete for a suitable throughput, which are used to transfer payload data. The latency of a message is then the sum of the latencies of each of its segments. So, the segment size should be large enough to accommodate one of these small messages.

There are two other considerations for the segment size trade off. One is that software implementations of a segmentation layer are very inefficient with small segment sizes. The other is that each segment or packet introduces an overhead in terms of header size and network switching time which decreases the overall throughput of the network. However, reducing the segment size reduces the message jitter, i.e. the variance of the

message latency. This is very useful for soft real time applications with arbitrary message lengths.

Therefore, the segmentation technique is a valid option to reduce the maximum latency, but the segment size should be carefully chosen depending on the network characteristics and traffic expected.

4.2.2.2 Dynamic Segment Size

The setting of a segment size is a difficult trade-off problem. The ideal would be to modify the segment size depending on the current conditions. For example, if a flow has no other flows competing for the same network resource, there is no need to segment large packets. Why should the segment size have to be fixed? The reassembly process can be done anyway with variable sized segments.

Small segments reduce latency of small high priority flows because a blocked segment has to wait until the other's segment flow is sent. The shorter it is, the shorter it takes. Therefore, the ideal would be that the low priority packet that blocks the high priority packet is split, allowing the higher priority packet to immediately continue its route. However, this is not straightforward, as a segmentation layer that keeps the routing information of the split packet should be implemented in each router. In any case the tail of the split packet would still be blocking the previous links. A solution is to keep the segmentation layer in the source and provide a backwards signal that travels from the conflicting link back to the source, signalling that the current packet being transmitted should be segmented immediately. Figure 4-9 shows the scenario described with the two possible solutions (a, b) and the time sequence (1,2,3...).

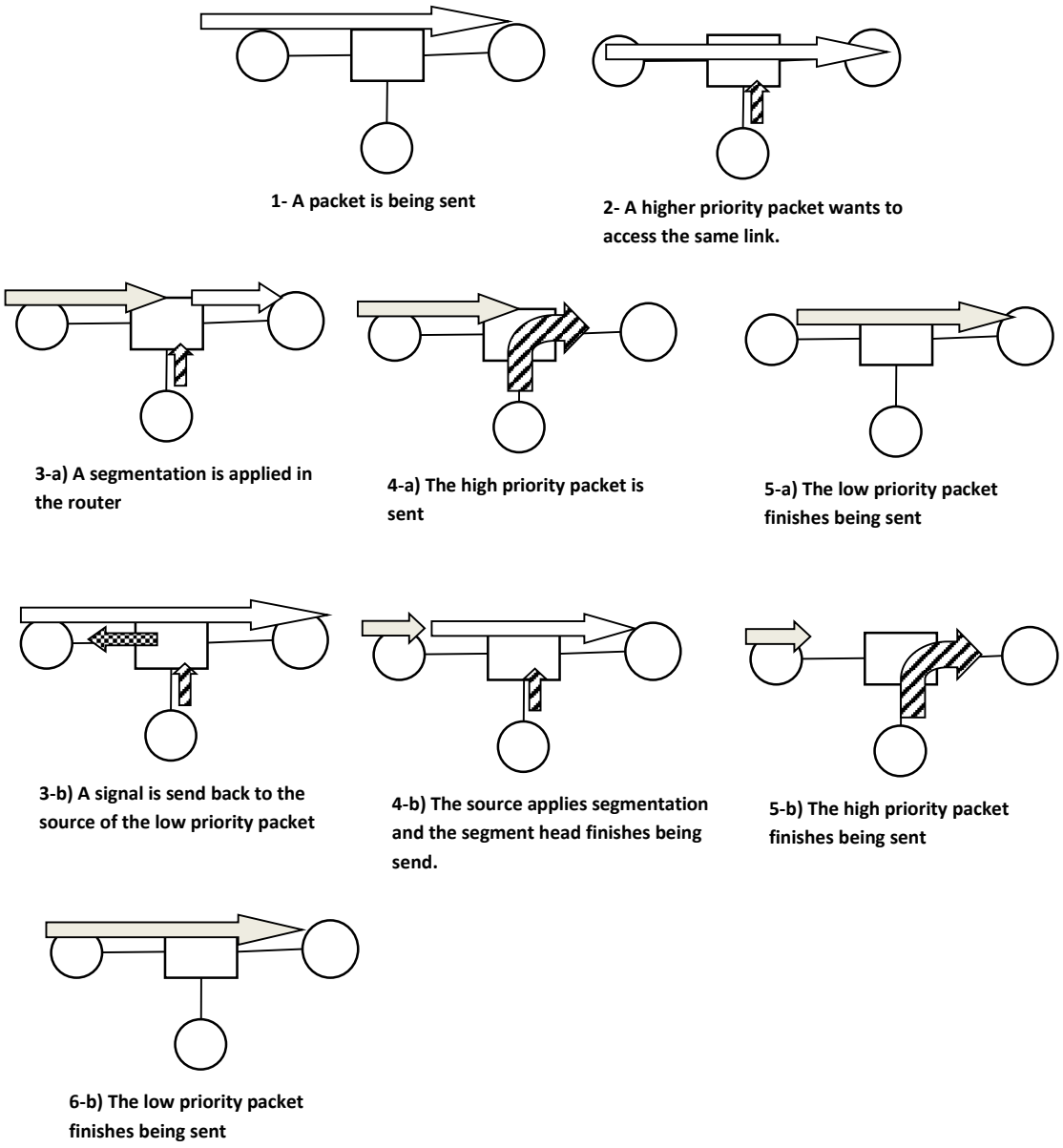


Figure 4-9: Sequence of events with dynamic segmentation mechanism

The reader could note that the previous example does not take into account the possibility that a packet may be blocked by another blocked packet. To cope with that case, the procedure can include the propagation of the notification signal up to the head of the blocked packet, and then to the tail of the packet that is not blocked. This signal should include information about the priority level of the blocked packet that has the highest priority.

The computation of an upper bound for the maximum latency is straightforward. The previous equation (4.3) for wormhole switching can be applied with a packet size or segment size equal to the buffer space in the routers along the maximum path, plus the propagation delay of the feedback signal described. This segment size is usually small as the buffers are small. The actual maximum latency will be much lower though, but it is difficult to quantify analytically.

4.2.3 Priority

Another common mechanism described in the background chapter to improve the Quality of Service is the use of different priorities for different data flows. In SpaceWire networks, the priority mechanism can be applied by the router when arbitrating between different data flows or can be applied at each sending node. The former case (priority applied by router) can reduce the latency of high priority flows as they do not need to apply, in equation (4.3), the summation term for each input port. The other case (priority applied by router), the sending priority determines which data flow is sent first when there are multiple pending data flows storing packets in one or more queues.

In section 4.2.1.2 it is assumed, for simplicity, that the latency value does not include the waiting time in the queues of sending nodes. The waiting time can be very high if the higher priority packets are not stored in a different queue than lower priority packets. In section 4.2.8.2 the impact of having multiple sending queues with different priority is evaluated with the help of the OPNET network simulator.

4.2.4 Congestion Control

Congestion control techniques try to reduce the latency and increase the throughput of some flows by reducing the injection rate, or traffic offered, by some lower priority flows

[Balakrishnan 2001]. Wormhole switching networks that have exceeded a saturation point, actually reduce the network throughput when the traffic offered increases. Also, in this scenario the latency increases exponentially and arbitrary buffer overflows can occur.

Therefore, it seems a good idea to reduce the traffic offered when unexpected high network congestion is detected. It should be applied to low priority flows with data rates that do not have time constraints. In order to have considerable impact on the congestion, the controlled traffic should be data flows with low priority and high data rate that use links that are network bottlenecks. The objective is to allow high priority flows to meet their latency requirements and avoid buffer overflow and data loss for flows that need a minimum throughput.

For example, the TCP protocol implements a congestion control algorithm by monitoring the packet error rate [Allman 1999]. It assumes that store-and-forward routers will drop packets when the network is congested and their buffers become full. Wormhole switching routers can also implement a timeout mechanism that spills packets if they are blocked for a long time. This approach requires a reliable transport protocol that can retry spilled packets and the assumption that other source of errors do not happen or are very unlikely.

However, with wormhole switching there is the possibility of performing a more efficient detection of network congestion. When a long packet is blocked, it will fill all the small router buffers across the network until the SpaceWire link flow control stops the transmission. Therefore, the sender can detect if a packet is blocked by monitoring the transmission time of packets. If the average packet transmission time unexpectedly increases, the source should pause sending packets for a certain time. The stopping period

should be random in order to support multiple nodes with this mechanism and avoid inefficient throughput cycles.

An even more precise technique could be used to monitor the link utilisation of each flow, instead of the destination path utilisation time. This would be much more efficient but it requires knowing the precise buffer space in the routers. Depending on the number of bytes already sent when the packet stops sending, it is possible to determine at which hop or link the packet is blocked. However, note that this technique cannot be used for bandwidth or link allocation, because the reserved bandwidth cannot be guaranteed. Still, it could be useful for limiting the use of a link by certain flows.

Besides the fact that packet injection rate control does not provide QoS guarantees, this technique may have little use for most current space applications, so it will not be used in the protocols developed in this thesis. High data rate flows produced by satellite instruments and sensors are constant data sources that require a minimum throughput and have small buffers. They cannot be stopped arbitrarily for unlimited periods to reduce the network congestion. There are other data sources though, like data compressing units using a mass memory unit, which could be suited for congestion control algorithms.

4.2.5 Virtual Channels

The most used solution to reduce the blockage effect in wormhole switching networks is to implement virtual channels [Dally 1992]. With this technique, when one packet is blocked, (because it cannot access an output link being used by another packet), the packet just blocks the buffer resource associated to the virtual channel used. It does not block the use of a physical link. So, the use of multiple virtual channels increases the throughput of the network and reduces the latency. However, some studies show that with basic round-robin virtual channel arbitration, in most networks, the network saturation

point only improves logarithmically with the number of virtual channels and more than five virtual channels do not give a valuable gain [Dally 2003].

Virtual channels can also improve the maximum latency, though it depends on the implementation. The assignment policy of virtual channels to different flows plays a critical role. Typically a virtual channel number can be allocated to a traffic class or priority level. A flow with a traffic class will only be able to access virtual channels assigned to the same traffic class, though usually there is only one virtual channel per class. Using priority levels, in some implementations a high-priority packet flow can also access one of the virtual channels assigned to lower priority flows. This last option may provide lower latencies for high priority traffic but is more complex to analyse and implement as it requires in the routers some complex logic to determine the destination virtual channel in addition to the destination port. In any case, the policy chosen and its benefits are tightly related to the arbitration mechanism used between virtual channels when accessing to the physical link, i.e the MAC implementation. In chapter 7 it is explained how SpaceFibre uses a simple one-to-one virtual channel mapping and an advanced MAC is defined to achieve the desired QoS.

Virtual channels are very useful for wormhole switching networks as they dramatically decrease the average latency and increase the throughput. The problem is that they are not compatible with the current SpaceWire standard, so they cannot be used in this type of network. Virtual channels could be supported in a future SpaceWire standard version following the feedback obtained with SpaceFibre. In chapter 7 it is explained how SpaceFibre uses virtual channels in the link layer to provide end-to-end virtual circuits and virtual networks.

4.2.6 Time-Division Multiplexing

In contrast with previous solutions, Time-Division Multiplexing (TDM) techniques can provide guaranteed maximum latency and guaranteed minimum throughput to specific flows. The key point is that they eliminate network congestion and its associated unpredictability. The average latency and throughput can be higher or lower, depending on the traffic shape and the scheduling used. One typical drawback is the difficulty to efficiently accommodate random traffic. Another problem is how to reuse unused reserved bandwidth for best effort traffic.

Networks resources, such as a set of links and their assigned buffers, are reserved to specific traffic flows during specific periods of time, called slots or timeslots. These slots are usually periodic and with the same duration. During each slot one or more packets can be sent. These slots may cycle with a higher period called epochs. They create a temporal virtual circuit that is usually set periodically using a schedule table. Other TDM techniques use alternative approaches not based on schedule tables, but they all have in common the use of a global notion of time.

One of the main advantages of TDM is that it can provide deterministic network behaviour which, in addition to giving strict guarantees in latency and throughput, makes it simpler to test the overall system when the network traffic is also constrained and predictable. In addition, the network design flow for scheduled networks can make use of constraint analysis to determine the system architecture based on the traffic expected and the QoS guarantees required. On the contrary, non-scheduled networks usually follow another design flow, in which the verification of the QoS constraints are verified after a system architecture is defined, which is modified if the requirements are not met. Figure 4-10 shows these different approaches.

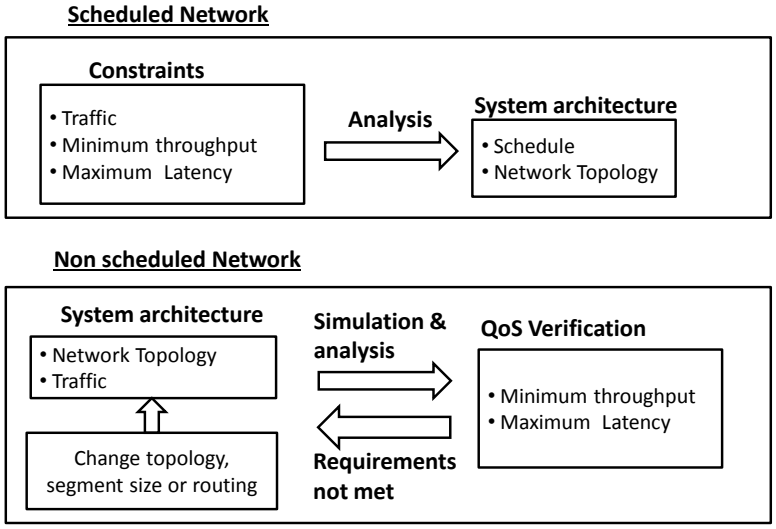


Figure 4-10: Network design flow

Figure 4-11 shows how TDM techniques can be classified as being centralised or distributed. The performance of TDM increases if multiple transactions or data flows are possible within the same timeslot. It also has more flexibility if the destination of each data flow can be determined by the data source at the beginning of each timeslot depending on the current state. These possibilities can be applied to both centralised and distributed categories.

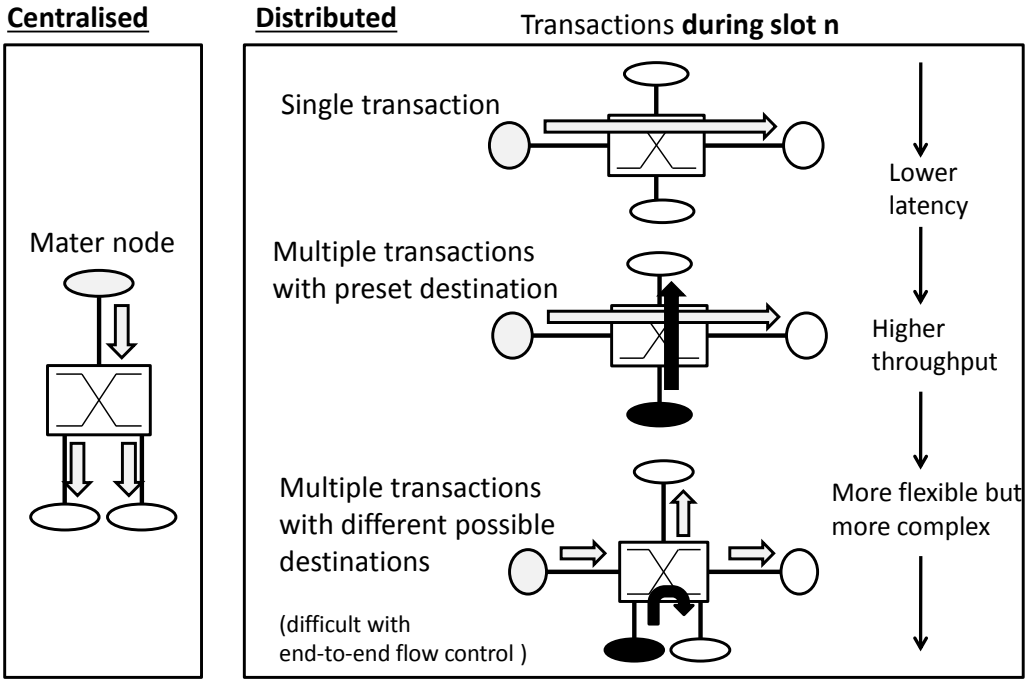


Figure 4-11: TDM scheduling architectures

4.2.6.1 Centralised

The simplest implementation of TDM is based on a single master node that completely sets and initiates all transactions of the network following a global schedule. Other nodes do not need to be aware of the time and the global schedule. Multiple concurrent transactions may be supported if they do not share the same network resources.

The advantage of this technique is that it is very easy to reconfigure the schedule, as it is not distributed. The drawback is that it requires periodic polling of every terminal to accommodate traffic that is randomly triggered by other terminals. Therefore it is a very simple solution that is only efficient if the data rate is low and the traffic is known and highly deterministic. This is usually the case for platform avionics systems. For other scenarios such as payload data-handling systems, a distributed approach is better suited.

4.2.6.2 Distributed

With distributed TDM, every node of the network is time-synchronised and can autonomously trigger a flow or transaction following a set of rules derived from a local scheduling table or a network arbitration mechanism.

A local schedule table decides the traffic generated by a node but it must be set up taking into account the network traffic of other nodes. Multiple packet flows can be active during the same timeslot but the same network resources, i.e. links, cannot be requested simultaneously by more than one packet.

Another approach is to use an arbitration mechanism to globally decide, per each slot, which packet flows of the network are enabled, based on network resources and priorities. In case of two flows using the same network resource, the highest priority would be granted. This procedure is analogue to the process of requesting a virtual circuit for some

period. The difference is that the setup time should be bounded and the duration of the virtual circuit depends on the duration of the Timeslots. A classic implementation requires an arbiter node that, at the beginning of a timeslot, receives all the requests and grants virtual circuits following a specific policy.

This approach is very well suited to unpredictable traffic but is highly inefficient for periodic high data rate traffic, because of the duration of the arbitration step during each slot. In contrast, the local scheduling is efficient for periodic traffic but can be problematic with random traffic. Space applications are usually set in a closed environment and the traffic is quite periodic and expected, so the local scheduling seems a better option. However, even if the traffic is known, it can be sporadic, which usually reduces the efficiency of the network.

So far only the arbitration and allocation of flows within the network has been accounted for. However, arbitration of flows within the same node should also be considered. More than one flow could be enabled during the same node and slot. They could go to the same or different destinations using paths that are allocated to this node and slot. The use of TDM guarantees that the selected flow will obtain a deterministic throughput. The arbitration scheme within the node could be priority based or bandwidth reserved and could improve the performance of the network when handling sporadic traffic.

4.2.7 Summary

Each SpaceWire network application scenario should select the solution that best suits its needs with minimum cost. Given, for each flow, a set of requirements in terms of maximum latency and minimum throughput, first an analytic analysis should be performed. If the traffic is mostly random and sporadic, mostly based on high data-rate flows without tight latency constraints, the segmentation mechanism would be the best

option. It will reduce the analytical upper bound of the maximum latency and increase the average throughput.

However, space applications are usually quite periodic and predictable. For these the best option seems to be TDM with a synchronous network using local scheduling. The nodes that are the origin of flows with sporadic traffic should implement local arbitration mechanisms. Table 4-2 shows a summary of advantages and disadvantages of each mechanism previously discussed.

Table 4-2: Summary of different QoS mechanisms

Method	Advantages	Disadvantages	Cost
Segmentation	Reduces the average and maximum latency for small messages.	Can increase the latency of long messages. Does not provide hard QoS guarantees	Medium
Priority	Reduces the average and maximum latency for high priority messages.	Does not provide hard QoS guarantees	Very low
Congestion control	Reduces the network congestion to obtain better average and maximum latency for particular messages.	Not suitable to most use cases in space applications where the traffic is known and expected and cannot be arbitrary reduced.	Low
Virtual Channels	Can decouple flows using different virtual channels. Increases link utilisation.	Not compatible with current SpaceWire standard. Requires more memory resources.	High
TDM	Provides deterministic delivery and guaranteed throughput and latency	Can be inefficient if traffic is mainly sporadic.	Medium

4.2.8 Experiment: SpaceWire Network Simulation

This chapter have pointed out the necessity of network simulations to obtain the average latency of different messages. This is especially important considering the advantages of using a non-deterministic QoS solution like the segmentation mechanism with packet sending priorities. As there was no simulation tool commercially available for SpaceWire

networks, a complete network model was developed using the popular OPNET tool. One objective was to study the impact of network congestion and the use of priority when sending packets. The other objective was to assess the improved average latency of control messages when segmentation was implemented.

4.2.8.1 OPNET Modelling

The OPNET tool allows the description of a network model at each different layer. First, the SpaceWire codec state machine was designed following the standard. The resulting state diagram is shown in Figure 4-12, where the red states are waiting states and the green states are used to create data packets.

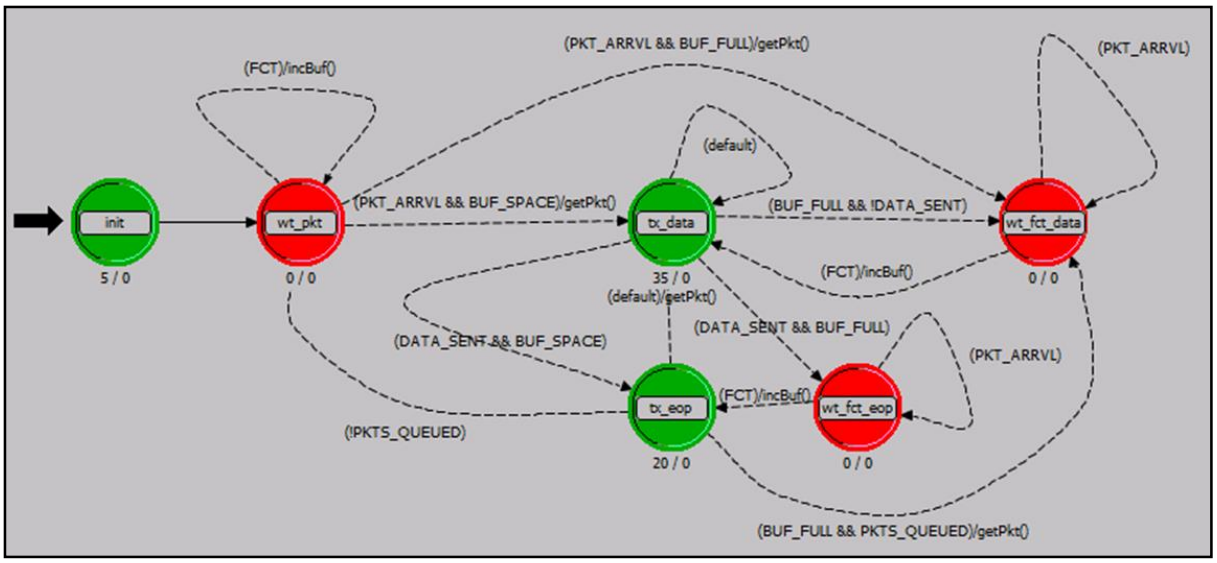


Figure 4-12: OPNET model of the codec state machine.

The next step was the design of the SpaceWire Node. To achieve this objective, two sources of packets were modelled, the *pktDataSource* and the *pktControlSource*, as shown in Figure 4-13. The *pktDataSource* creates packets of 4 Kbytes and the *pktControlSource* packets of 256 bytes.

Figure 4-14 shows the router model which was designed with 8 ports. The OPNET model was validated by running simple scenarios that produced results which could be compared with analytical calculations.

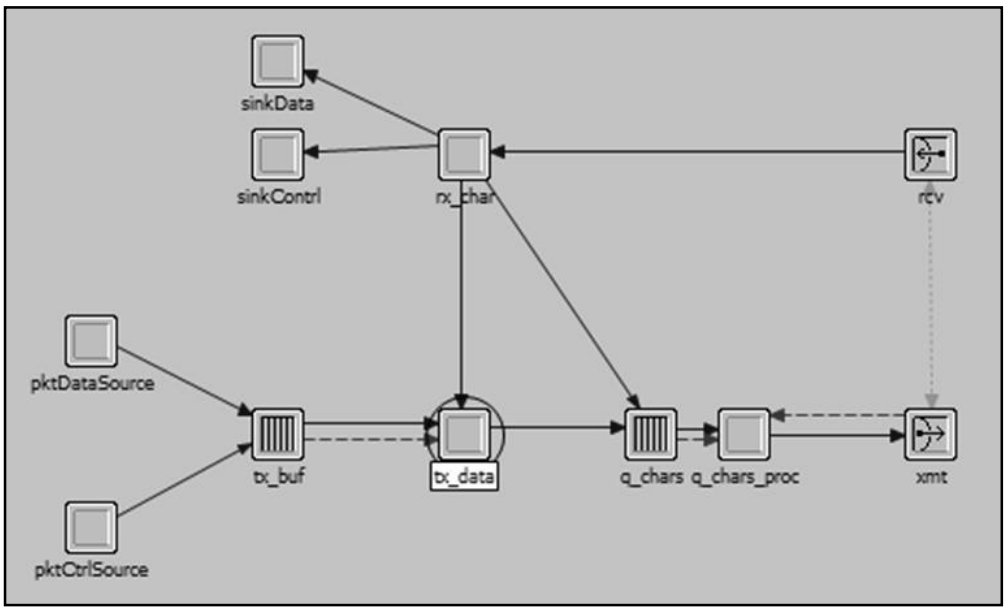


Figure 4-13: OPNET node model of the codec state machine.

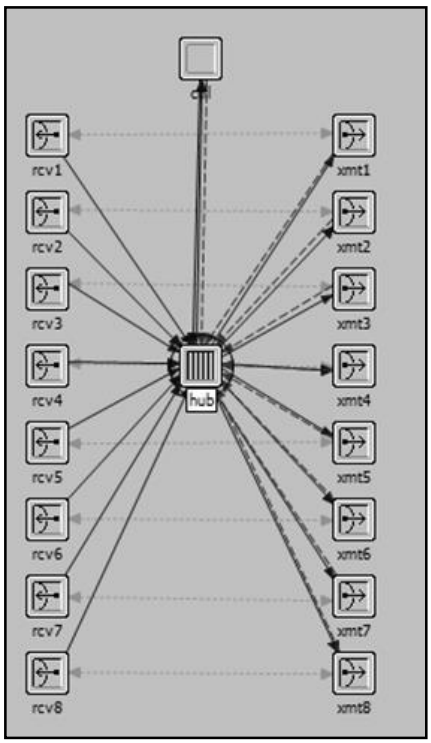


Figure 4-14: OPNET router model.

4.2.8.2 Results

This section presents the results of OPNET simulations and the analysis of these results using analytical models. This will provide a better understanding of the benefits of using priorities and segmentation techniques.

4.2.8.2.1 Single Sending Queue

In this first scenario, one node sends control and data packets to a destination using a single common packet queue within the node, as shown in Figure 4-15.

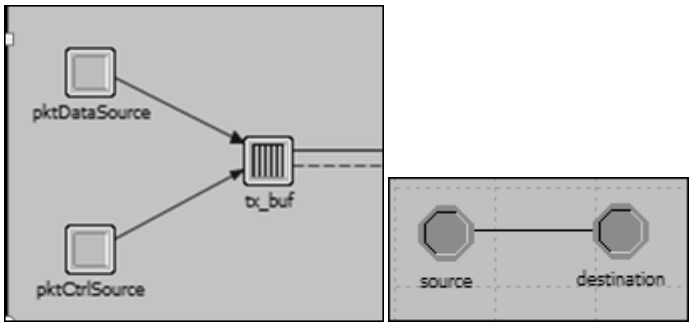


Figure 4-15: OPNET node model with one queue (left). Network topology (right)

The network traffic is detailed in Table 4-3, where the OPNET term of inter-arrival time is defined as the average time elapsed between two consecutive packets being generated. The control packets are much smaller than the data packets and the total data rate is adjusted to use around 20% of the bandwidth of the link (10Mbit/s). Note that with SpaceWire, one byte of user data requires the transmission of a SpaceWire data character of 10 bits.

Table 4-3: Network traffic for OPNET simulation

Packet type	Packet size	Tx time @ 10Mbit/s	Inter-arrival time	Data rate	Link utilisation @ 10Mbit/s
Data Packet	4000 bytes	4ms	20ms	2 Mbit/s	20%
Control Packet	256 bytes	0.25ms	10ms	0.256 Mbit/s	2.56%

Figure 4-16 shows the OPNET statistic plot for the average latency for both types of packets. The data packets average latency (end-to-end delay) converge in the graph to around 4.5ms and the control packet latency converge to around 0.75ms. They are both 0.5ms higher than their respective transmission time (4ms and 0.25ms). This additional delay is mainly due to the waiting time in the queue when there is a data packet being transmitted. The arrival of packets is configured to be not deterministic and follows a Poisson or exponential distribution. Therefore, it is possible that two packets are generated in a very short period, so the second one has to wait until the first one has finished being sent.

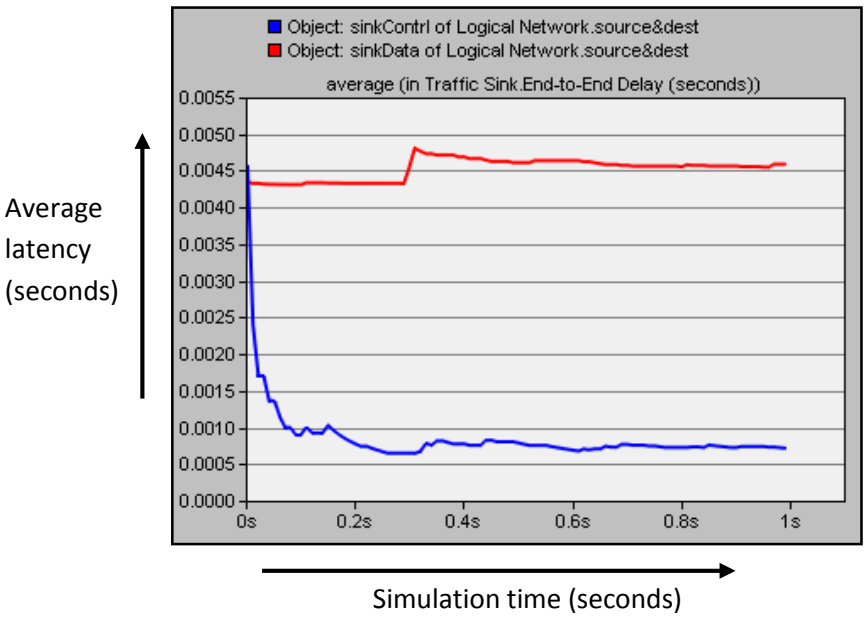


Figure 4-16: Average latency of data and control packets obtained with OPNET

The values measured can be obtained theoretically using basic queuing theory [Sundarapandian 2009]. Equation (4.7) gives the average response time $\overline{r}$ (transmission or service time T_{tx} plus queuing delay T_{wait}) in a system with the Kendall's notation M/D/1, which corresponds to an exponential packet inter-arrival time with a deterministic service time and one server.

$$\overline{r} = \frac{1}{\mu} + \frac{\lambda}{2\mu^2(1 - {\lambda}/{\mu})} = T_{tx} + T_{wait} \tag{4.7}$$

Note that λ is the inter-arrival rate (inverse of the inter-arrival time) and μ is the service rate (inverse of the transmission time T_{tx}). Using the parameters of Table 4-3 the response time or total delay is obtained in the case where only data packets and only control packets are sent.

$$\overline{r}_{data} = \frac{1}{250} + \frac{50}{2 * 250^2(1 - {50}/{250})} = 4 + 0.5ms \tag{4.8}$$

$$\overline{r}_{ctrl} = \frac{1}{3906} + \frac{100}{2 * 3906^2(1 - {100}/{3906})} = 0.25 + 0.003ms \tag{4.9}$$

It can be seen that the waiting time in the queue obtained with OPNET (0.5ms) matches the second term of equation (4.8). This indicates that this waiting time is mainly due to data packets being send or stored in the queue, and the impact of control packets is so small that it is not noticeable.

It is interesting to use equation (4.7) to see the impact in the data packet latency when increasing the rate of the data packets (λ) and therefore the link utilisation. Figure 4-17 shows the plot of the average data packet latency as a function of the link utilisation.

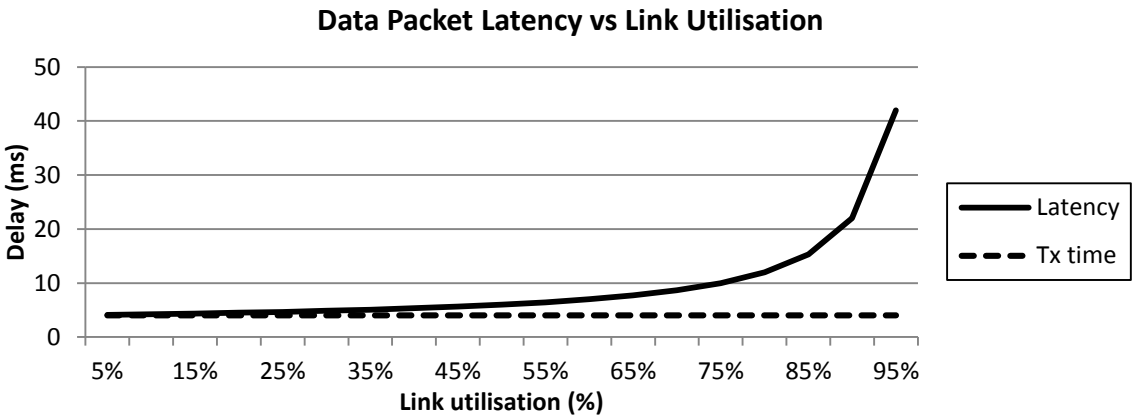


Figure 4-17: Average latency of data packets versus link utilisation

Note that if a constant deterministic inter-arrival packet time is used instead of an exponential one, the data packet latency will be equal to the transmission time, as there will never be a data packet waiting to be transmitted.

It is also worth noticing that basic queuing theory also provides the average number of packets $\overline{n}$ in the queue using Little's law.

$$\overline{n} = \lambda \overline{r} \tag{4.10}$$

4.2.8.2.2 Two Sending Queues with Different Priority

It has been shown that the latency of control packets can be improved using a different priority when sending each packet type. If the control packets are being sent with higher priority and they use an independent sending queue, the control packets do not need to wait for other data packets of the queue to be sent. A new generated control packet has to wait only if there is a data packet currently being transmitted or there are other control packets waiting in the control packet queue (see Figure 4-18).

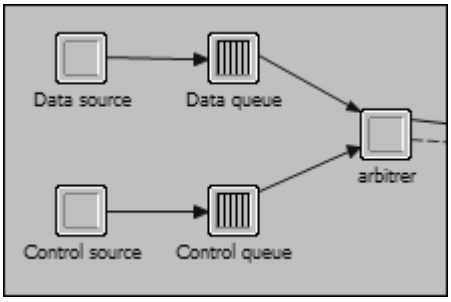


Figure 4-18: Multiple packet queues in an OPNET model

Multiple queues with different priorities can be easily simulated without modifying the single queue OPNET node model developed and validated, using a SpaceWire network with two nodes and some restrictions on the traffic generated.

- A single node with two different queues, but with the same priority, is equivalent to two nodes with a single queue each one, connected to a router, as shown in Figure 4-19. The SpaceWire router arbitrates between different nodes using a fair round-robin mechanism.
- In order to obtain, with this OPNET model, the average latency of control packets when they have higher priority than data packets, it is ensured in the simulator that the control queue never holds more than one control packet. So, it is not sampled the particular case when a second control packet waiting in the control packet queue must wait for a data packet being transmitted plus the transmission time of the first control packet plus the transmission time of the next data packet. This case occurs using the OPNET model of SpaceWire router with round-robin arbitration and does not occur in a model of two queues with different priority competing for the same link.

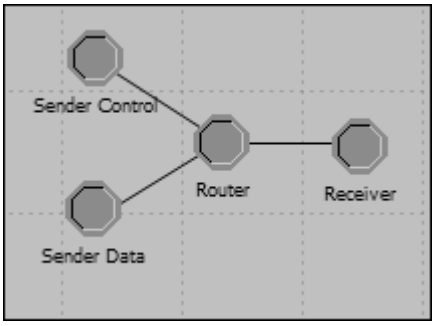


Figure 4-19: Network topology alike to a node with two queues with equal priority

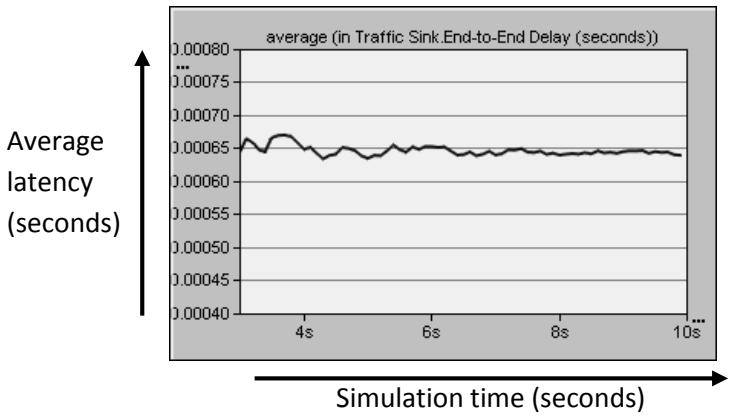


Figure 4-20: OPNET average control packet latency

Figure 4-20 shows that the average latency of control packets is 0.65ms instead of the 0.75ms obtained without using a dedicated higher priority queue for the control packets. This value can be obtained analytically if it is considered that in the described scenario, the average waiting time of a control packet is the probability that the link is transmitting a data packet multiplied by the average time the data packet needs to finish being transmitted. The first term is the link utilisation and the second term is half of the transmission time of a data packet. When a control packet arrives while a data packet is being transmitted the number of remaining data bytes is uniformly distributed. So, the worst case is that the second term is equal to the transmission time, and the best case is that the second term is zero, so on average the value is half the transmission time. Finally, the total response time is the sum of the transmission time plus the waiting time described, and it is shown in equation (4.11), where S is the link speed and L is packet length.

$$\overline{r}_{ctrl} = \frac{1}{\mu_{ctrl}} + \frac{\lambda_{data} * L}{S} * \left(\frac{1}{2} * \frac{1}{\mu_{data}}\right) = \frac{1}{3906} + \frac{50 * (4000 * 10)}{10000000} * \left(\frac{1}{2 * 250}\right) = 0.65ms \quad (4.11)$$

The benefits of using the priority mechanism increase when the link utilisation is higher. Figure 4-21 compares the control packet latency when using (or not) the priority mechanism depending on the link utilisation. The graph is obtained with equation (4.11) using λ_{data} as a parameter. It shows that when using priorities, the latency of sporadic control packets increase linearly with the link utilisation instead of increasing exponentially.

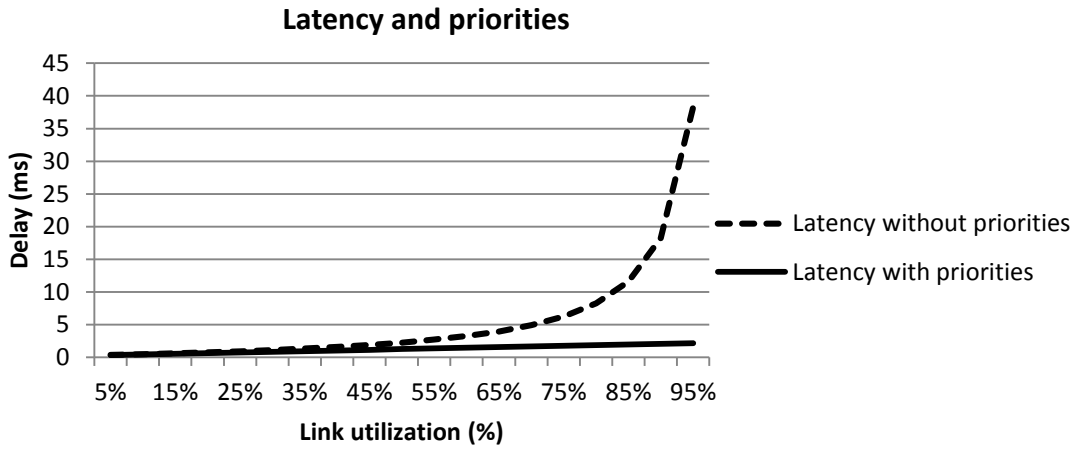


Figure 4-21: Average latency of sporadic control packets with high priority

The latency of control packets can be further reduced using the segmentation mechanism described in this chapter. The large data packets are sent in multiple SpaceWire packets containing segments of the data message. Therefore, in equation (4.11), the data packet length *L* can be much smaller.

Figure 4-22 shows the latency of control packets using priorities and different sizes of segments. Note that the average latency of a data message is the average latency of a packet containing a segment multiplied by the number of segments required.

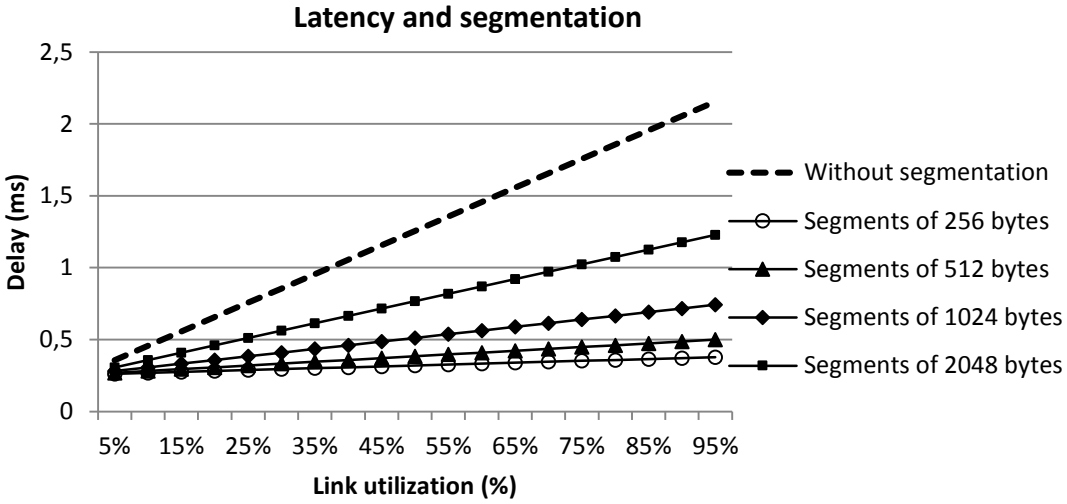


Figure 4-22: Average latency of control packets using priorities and segmentation

4.2.8.2.3 Two Nodes with one Sending Queue each

So far the use of priorities and segmentation have been analysed when there is a single node with multiple queues.

Now the case of two different nodes with one sending queue each will be studied. One node sends data packets and the other sends control packets with the topology shown in Figure 4-19 and Figure 4-23. Therefore, to obtain the experimental results, it is only required to run the last described OPNET simulation without enforcing that the queue in the node sending control packets can hold only one packet.

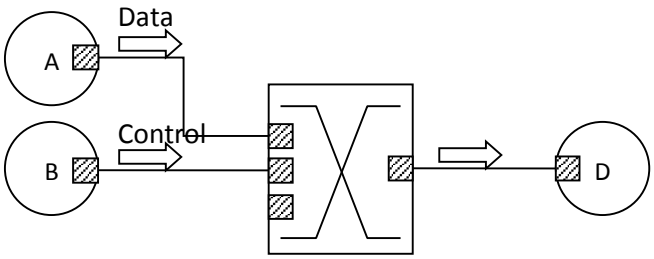


Figure 4-23: Network topology

The control packets are generated with an exponential distribution as in a real scenario they are usually not sent with a deterministic periodicity. The data packets are generated with both exponential and a constant inter-arrival time. This last case represents the case of an instrument which generates data at a constant known rate. The data messages are not segmented to simplify the comparison of results with the previous setups. Figure 4-24 shows the OPNET results and it also includes the previous results without segmentation of Figure 4-22.

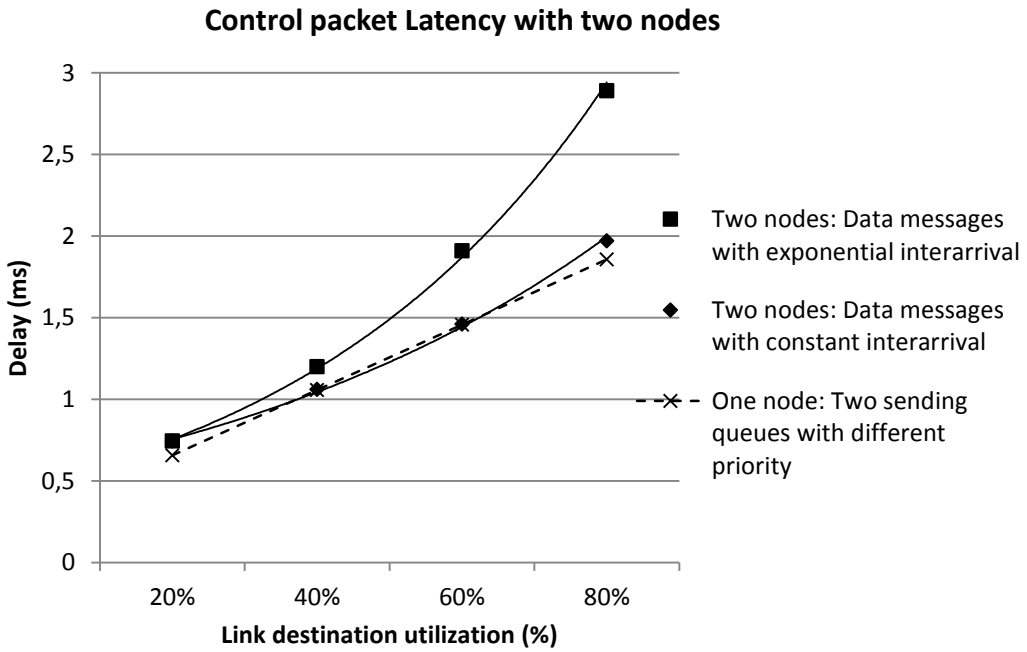


Figure 4-24: Average control packet latency in a network with one competing node

The simulation shows that the values obtained with two nodes with data messages generated with constant inter-arrival time are very similar to the ones obtained in the

previous case of one node with two sending queues with different priority. Therefore, equation (4.11) provides also a good approximation in the current case of two nodes if the data messages are generated with a constant periodicity. The explanation is that, in this scenario, if the data messages are periodic, the probability that the packet queues in the data and control node hold more than one packet is very low, so having (or not) a priority mechanism in the router does not have much impact. As arbitrating between two nodes with one sending queue each is equivalent to arbitrating between two sending queues in one node, and the priority mechanism does not matter much, in this scenario the model of two sending queues can be applied with different priorities and the results of Figure 4-22 can be used as a good approximation.

An interesting remark is that Figure 4-24 makes clear that if there is an instrument that produces data packets at a constant rate, then the latency of a control packet being sent from the other link will be lower than if these data packets are produced with an exponential distribution. The worst case control message latency is also much better as shown in the histograms of the control packets of Figure 4-25, which were generated at 80% link utilisation with a deterministic and exponential data packets inter-arrival time.

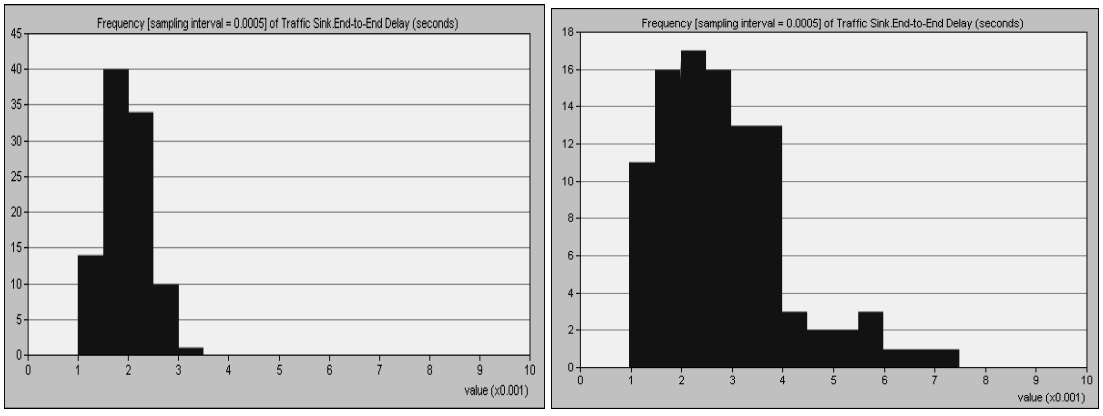


Figure 4-25: Latency histogram for constant (left). Exponential (right) arrival time

4.2.9 Experiment: Wormhole Switching Custom Simulator

Section 4.2.2.2 presented the idea of dynamically modifying the segment size depending on the current congestion of the network. It proved difficult to implement such technique using the OPNET simulator, so a custom network simulator was developed that focussed on evaluating this novel technique.

The code was written in a high-level general-purpose language and it only simulated the network elements required for the evaluation of the dynamic segment size technique. Simplicity and the capability to easily observe the operation of the simulator was more important than the performance obtained by commercial or dedicated network simulator solutions. Besides, the OPNET simulator is focussed on store and forward packet network architectures while the one developed was only for wormhole switching networks like SpaceWire.

A very simple graphical interface was developed to show the global network operation. Figure 4-26 shows a screenshot of the simulator tool developed while doing an animation of packets moving across the network during a particular simulation. The rectangles are routers that contains buffers holding one or more flits (8 SpaceWire characters) shown as small light squares.

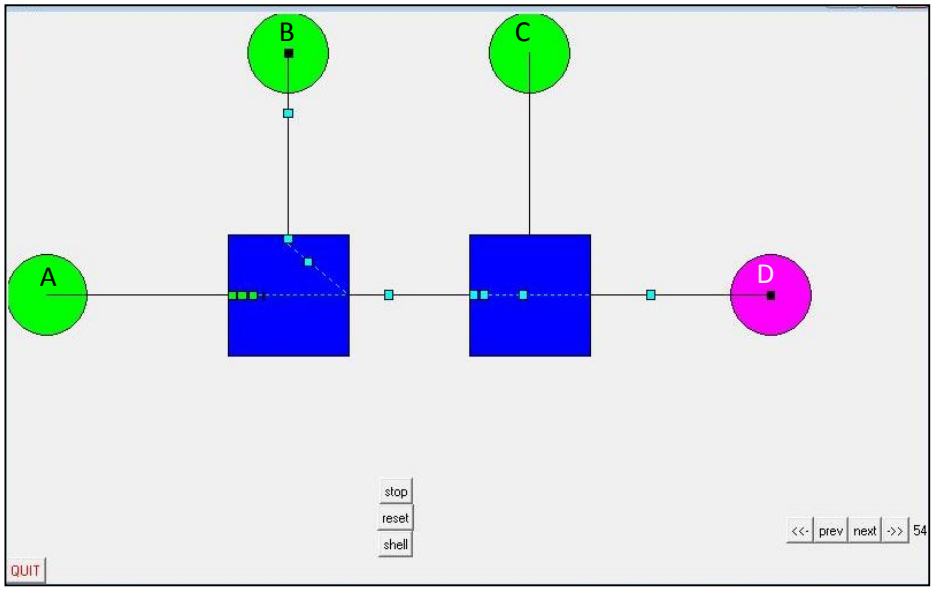


Figure 4-26: Screenshot of the custom simulator developed

This wormhole switching simulator featured the option to implement the dynamic segmentation technique as a simulation parameter.

4.2.9.1 Results

The network topology used corresponds to the same one as shown in Figure 4-26. The three lighter nodes A,B,C send both data and command packets (with higher priority) to the dark node D. Figure 4-27 shows the latency and throughput of each packet flow depending on the overall traffic offered or injected to the network, which is evenly divided between all flows. The source of each flow is represented by a circle (node A), a square (node B) and a triangle (node C) and they are divided between control and data packet. Note that nodes A and B are further away from the destination D than node C. Results clearly show that the distance to the destination affects the QoS metrics. Latency and throughput from nodes A and B are worse than node C which is very characteristic of wormhole switching networks.

In contrast, Figure 4-28, which illustrates wormhole switching with dynamic segmentation shows how each flow has a similar latency independent of the distance to the destination. This is the result of applying an initial implementation of the dynamic segmentation technique following the description of section 4.2.2.2 .The nearest node increases its latency and the further nodes decrease the latency. For example, when the traffic offered is 30% the latency of the control packets of node A,B is reduced from 140μs to 90μs approximately.

However there is a high cost for node C, the nearest node to the destination, especially for the throughput of data packets when the traffic offered is high. This happens because its packets are periodically being segmented each time a higher priority packet comes from nodes A and B. Also, the overall network throughput is reduced due to the overhead introduced by the dynamic segmentation technique when using realistic delays in the simulation parameters. Therefore it is questionable whether, with the current implementation, the proposed idea can provide significant benefits and it is left to future work to investigate further the possibilities of the dynamic segmentation technique.

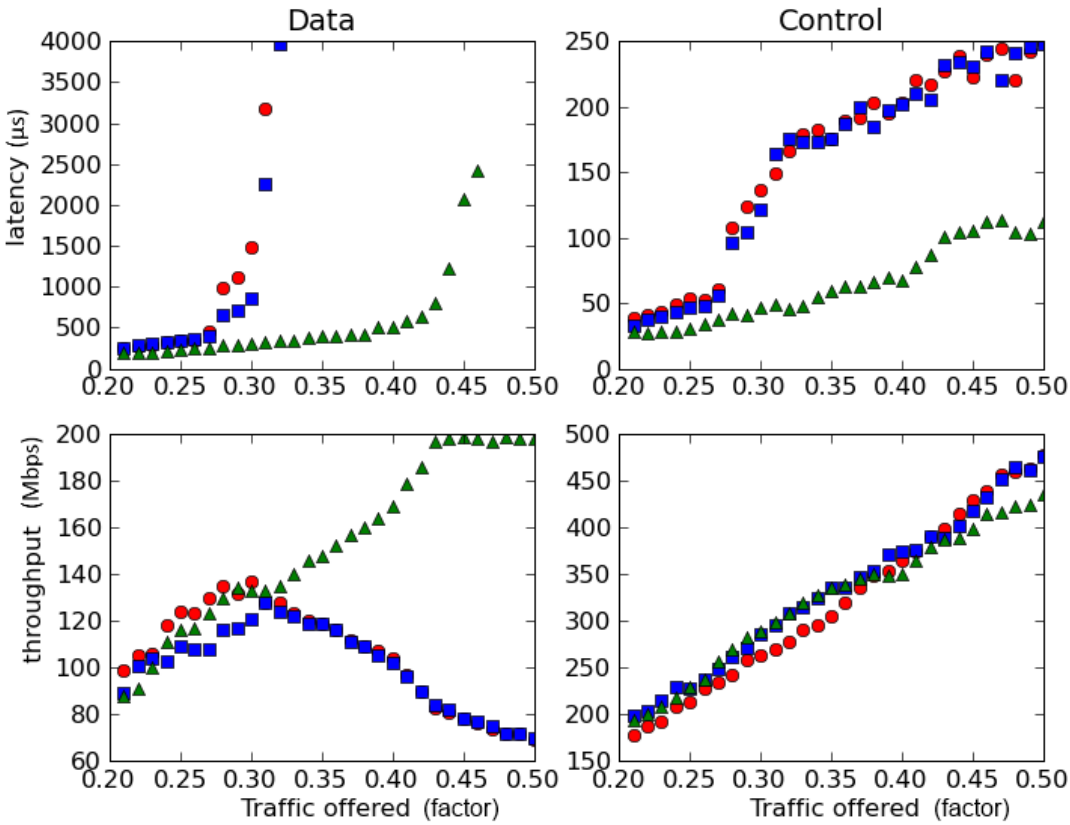


Figure 4-27: Wormhole switching with static segmentation

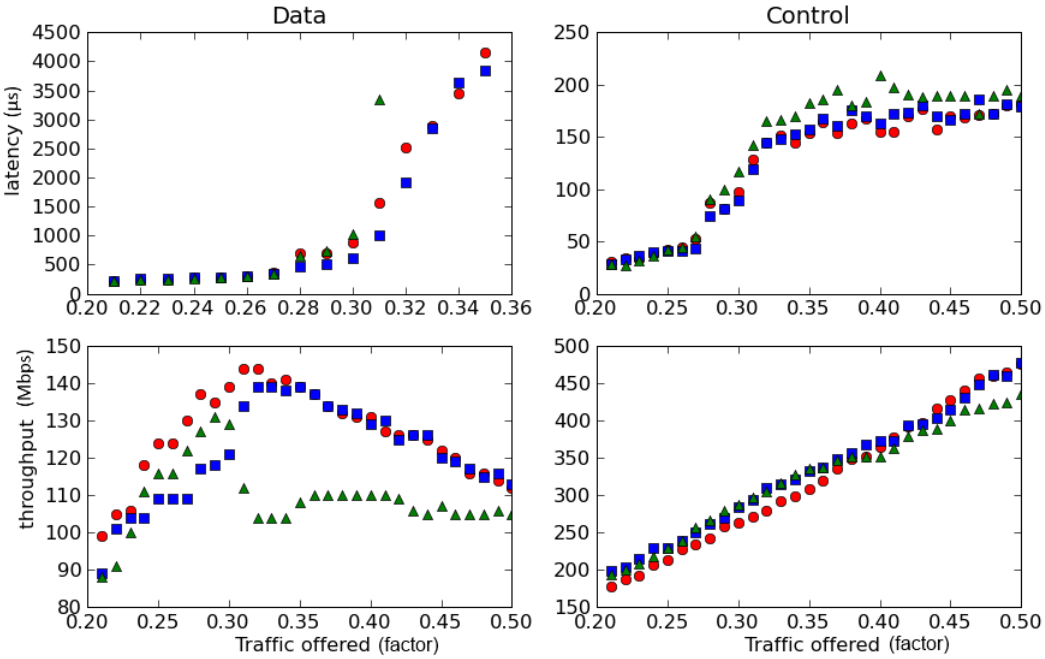


Figure 4-28: Wormhole switching with dynamic segmentation

4.2.10 Experiment: Real-Time Capabilities of SpaceWire Devices

This chapter has explored different solutions to provide real-time communication that makes certain assumptions on the capabilities of the network. It is one of the aims of this thesis to implement QoS solutions and methods that can work with the current generation of SpaceWire devices. Therefore, it is important to assess their capabilities, specifically the ones related with real time communication, such as congestion handling and time distribution using Time-Codes.

4.2.10.1 Congestion Handling of SpW-10X Router

The SpaceWire standard does not specify what to do when a packet is blocked due to congestion. However, most popular SpaceWire routers such as SpW-10X implements a timeout mechanism that spills packets that are being blocked during a certain period as shown in Figure 4-3. Note that in the particular case of SpW-10X router, the packet also waits during a certain time when the output link is not active, in order to deal with sporadic link disconnections.

This timeout feature can be an issue when using TDM mechanisms and synchronous networks. With TDM it is possible to configure the scheduling in order to avoid any congestion or switching arbitration between multiple packets from different input ports going to the same destination port. However, in case of an error, a packet may use a network resource, i.e. link, that was not previously allocated for it. If this packet is not removed from the network it may produce congestion, which due to the characteristics of wormhole switching may lead to more congestion in other parts of the network. Therefore, the default timeout mechanism of SpaceWire routers is not the best solution, as it does not remove the packet until the timeout elapses, which could take longer than the duration of a timeslot.

A better solution is to immediately spill a packet when it becomes blocked. The error is contained and the reliability QoS layer of a transport protocol can deal with the lost packet, using for example, a retry mechanism. Fortunately, after some analysis and experimentation, it became clear that it is possible to easily configure the SpW-10X router to deal with blocked packets in the previous way. The solution relies on the use of one or two of its external ports to be used as a data sink for a blocked packet.

The SpW-10X router should be configured as follows:

- The external port signal named “EXT10_OUT_READ_N” corresponding to port 10 or/and the external port signal "EXT9_OUT_READ_N" corresponding to port 9 should be set low.
- The routing table should be set to route all used logical addresses to the external port(s) used as a data sink. This is done by setting the appropriate bit(s) of the request field of the GAR registers.

4.2.10.2 Timecode Distribution

Time or tick network distribution is essential for TDM mechanisms. As previously explained, SpaceWire provides low-latency link-layer characters called Time-Codes, that are broadcast and sent within data packets across the whole network.

This experiment involved the measurement of the precise latency and jitter of broadcast distribution across each router. The results matched the specifications for the router analysed, the SpW-10X. This means that when running the link at 200Mbit/s it is possible to obtain time accuracy of less than one microsecond for small networks distances of one or two routers. Larger networks increase linearly the latency and jitter of Time-Codes.

Another aspect is the reliability of the Time-Code transmission. This character as any other SpaceWire link-layer characters is protected only by a parity bit. A parity bit should, in general, detect any error produced by a single glitch in the link, but in case of Data Strobe encoding, this may not always be the case. For example, a glitch in the strobe line could cause one or two bits to be duplicated in the resulting data stream, producing more than one bit modification. This is likely (but not guaranteed) to produce a parity error at some point later on, but before this is detected, some wrong characters could be considered valid.

A software analysis was developed to simulate this kind of errors and see possible undesirable outcomes. Results showed that indeed the parity bit was not enough to cover single error glitches. The outcome is dependent on the data pattern tested, but an interesting error case was observed. Figure 4-29 shows the NULL and the Time-Code character. Null characters are sent when the link is not sending user data, so it is quite usual that a link is only sending these characters. The Time-Code character is quite similar and it is easy to see that a duplication of one of the first three bits after the parity bit, can produce a valid Time-Code, with a value of 142, if nulls are sent by the transmitter. After this spurious Time-Code is received with a valid parity bit, the next character produces a parity error. This valid Time-Code will not be distributed over the network if the previous Time-Code was not 141.

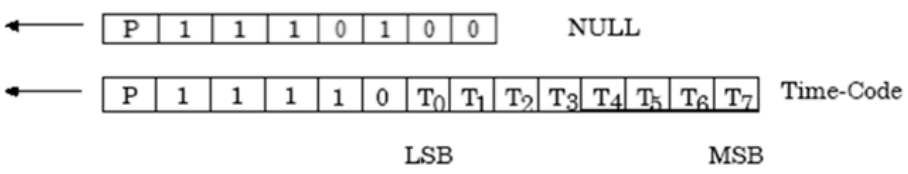


Figure 4-29: SpaceWire NULL and Time-Code characters

Therefore, it is desirable that the tick distribution of TDM timeslots should not completely rely on the robustness of Time-Codes characters reception so the system should include

a redundant mechanism such as the local timeslot prediction implemented in protocols presented in the following chapters.

4.3 Conclusions

This chapter has analysed the best techniques for the implementation of the two main QoS aspects, reliability and timely delivery, to existing SpaceWire networks.

In addition, a new solution to compute the upper-bound packet latency for wormhole switching has been presented and the following experiments have been performed:

- Implementation and evaluation of the SpaceWire-RT protocol
- OPNET simulation of a SpaceWire network for the evaluation of the benefits of segmentation and priority mechanisms.
- Network simulation of a novel segmentation mechanism
- Evaluation of the Time-Code distribution mechanism to evaluate the capability of SpaceWire scheduled networks.

Results of this work will be used in the definition of the protocols defined in the following chapters.

Chapter 5: Transport Layer Quality of Service for SpaceWire

This chapter describes how to implement a Quality of Service layer for SpaceWire networks using a protocol defined at the transport layer. For this purpose, two complete transport protocols are designed and prototyped using existing SpaceWire devices, taking into account the analysis performed in the previous chapter.

From the results of the experiment described in section 4.1.4 it can be concluded that the use of bidirectional channels can greatly increase the efficiency of the protocol when the data flow between two nodes is bidirectional. This will be the basis of the first protocol presented, the Bidirectional Transport Protocol (BTP), which is prototyped in software for Electrical Ground Support Equipment (EGSE) units.

The BTP protocol provides real-time characteristics using segmentation and scheduling. However, experimental results show that the scheduling mechanism induced some inefficiencies in certain network configurations that could be better solved with a new protocol optimised for scheduled unidirectional data flows. Feedback from the SpaceWire Working Group showed special interest in a synchronous protocol that gives more guarantees for command and control messages, indicating that this is a significant use case for a transport protocol. Therefore, a second protocol, the Unidirectional Transport Protocol is designed and prototyped in a radiation hardened component, the Remote Terminal Controller (RTC).

The next two sections describe each protocol explaining the design trade off, the protocol specification, the experimental apparatus, and its evaluation.

5.1 Bidirectional Transport Protocol

The Bidirectional Transport Protocol (BTP) aims to provide a transport layer for SpaceWire networks that offers Quality of Service regarding reliability and timeliness. It was built considering the specification of initial draft of the SpaceWire-RT [Parkes 2008d] and the results of its prototyping effort are described earlier in section 4.1.4

5.1.1 Requirements

The main requirement of BTP is to provide QoS to SpaceWire networks with the highest performance. BTP is the first protocol developed by the author of this thesis so in this section generic guidelines are presented that are also applicable to other protocols developed in other sections.

5.1.1.1 Objectives, Guidelines and Use Cases

Before a set of protocol requirements are defined it is quite useful to consider more generic design guidelines and objectives. They are summarised below:

- High performance.
- Simple to implement and simple to understand.
- Reliable and robust.
- Flexible enough to accommodate multiple use cases.
- Enforce good network design practice that allows reduction of the protocol complexity. In other words, the intelligence is in the network configuration, not in the protocol.

- Support software implementations which have low processing power but a high quantity of memory.
- Support hardware implementations which have high processing power but a low quantity of memory.
- Support for processor-based intelligent nodes and slave nodes without a CPU interfacing instrument and other data source devices.
- Compatible with most significant SpaceWire-compliant devices.

This specific protocol should also be flexible enough to cover the following generic use cases:

- Asynchronous communication with dedicated links. It may require high throughput, reliability and end-to-end flow control.
- Asynchronous communication using shared links that tolerates variable message latency. It may require reliability and end-to-end flow control.
- Synchronous and reliable communication for periodic status messages.
- Synchronous and reliable communication for sporadic control messages, with opportunistic use by other types of messages of otherwise unused timeslots.

5.1.1.2 List of Requirements

The BTP must provide communication channels with:

- Data integrity, i.e. data delivered should be correct.
- Reliable data transfer.
- End-to-end flow control.

- Sending priorities.
- Segmentation for synchronous and asynchronous networks.
- Scheduling for synchronous networks.
- Suitable configuration mechanisms

Some capabilities like reliable data transfer and scheduling may be disabled by the user if their use implies a significant performance overhead and the additional capability is not required. However, most of them have been considered essential in the analysis of chapter 4 to provide QoS. For example, scheduling is considered the simplest way to achieve strict timing guarantees. Scheduling also requires implementation of segmentation capabilities which, if used without scheduling, still provide better latency figures.

5.1.1.3 End-to-end Flow Control

One question that may arise is why end-to-end flow control is a requirement if most space applications are very constrained in their expected operation and a destination buffer overflow would only happen when there is an error, in which case the reliability capability would be enough. The answer is that some applications not only rely on the use of packet reception buffers but they also rely on the availability of other resources. These resources may work on a message-by-message basis, in which case the resource is locked when the first segment of a message is received and remains locked until the last segment is received. The clearest example is a hardware implementation of the RMAP protocol.

The following scenario can be considered. There is a sender with two channels that allows sending of data to a RMAP handler at the destination. One channel has higher sending priority than the other, but the lower priority channel has data available first so this data is sent first. The problem appears if the high priority channel has data to send before the

low priority channel has finished sending all the segments of the message. The first segment of the high priority channel will not be accepted at the destination because the RMAP handler will be busy and not ready. The sending node may then try to resend the discarded packet without success. The low priority segments may not have a chance to unlock the RMAP handler because the channel that is retrying has higher priority.

The end-to-end flow control deals automatically with that issue. When the RMAP handler is busy, it does not read out data from the packet reception buffers. When they become full, the data source is notified and resumes sending the low priority packet segments until the RMAP handler resource is unlocked.

5.1.2 Design Considerations

This section analyses the basis of the BTP specification and performs a trade-off of different possibilities. Here it is assumed that the SpaceWire-RT description in section 2.3.1.2 and the SpaceWire RT prototype described in section 4.1.4 have been previously read. The following BTP description mainly focuses on possible improvements over the SpaceWire-RT protocol previously prototyped.

Table 5-1 gives an overview of the issues and modifications to SpaceWire-RT discussed in the following subsections.

Table 5-1: Overview of BTP improvements over SpaceWire-RT

SpaceWire-RT issue	BTP solution
The high inter-packet delay of SW implementations reduces throughput	Bidirectional channels and piggybacking
Small segment size reduces throughput in SW implementations	Maximum segment size configurable.
Lack of protocol initialisation	Reset channel mechanism
Maximum of 256 channels in the network	Channel number is node unique, not network unique.
Return path address must be configured using another protocol.	Return path address is included in each packet
Scheduled mode needs to allocate time in each timeslot for the acknowledgments to be received	The acknowledgments are received in the following timeslots.
A network link analyser cannot monitor the different QoS of each packet.	Each packet sent contains information about the QoS of the data flow.

5.1.2.1 Bidirectional Channels

The most important modification is the implementation of bidirectional channels using the piggybacking technique. This improves the efficiency of the protocol by reducing the impact of the delay introduced when sending each packet, called inter-packet delay. The fewer packets to send the more efficient is the protocol, especially for software implementations. Instead of sending separate packets for the data, the acknowledgment, the flow control and the acknowledgment of the flow control, all information is provided in a single packet header structure and it is sent even if some fields do not provide new information. Figure 5-6 shows the unified packet format or PDU used by the BTP.

Bidirectional channels are especially efficient for application protocols that are bidirectional like RMAP. Figure 5-1 shows the flow of packets between the RMAP command node and the RMAP target node when an RMAP read transaction is executed. Without piggybacking, a single transaction requires sending eight packets. With

piggybacking two RMAP transactions can be executed using only six packets. This is a significant reduction in the number of packets used, so the RMAP transaction latency due to packet processing delay is greatly reduced.

Piggybacking can also be justified from the fact that the original SpaceWire-RT protocol [Parkes 2008d] is intrinsically bidirectional even for unidirectional data flows. The source buffer sends data, the destination buffer sends flow control information, and both require independent acknowledge.

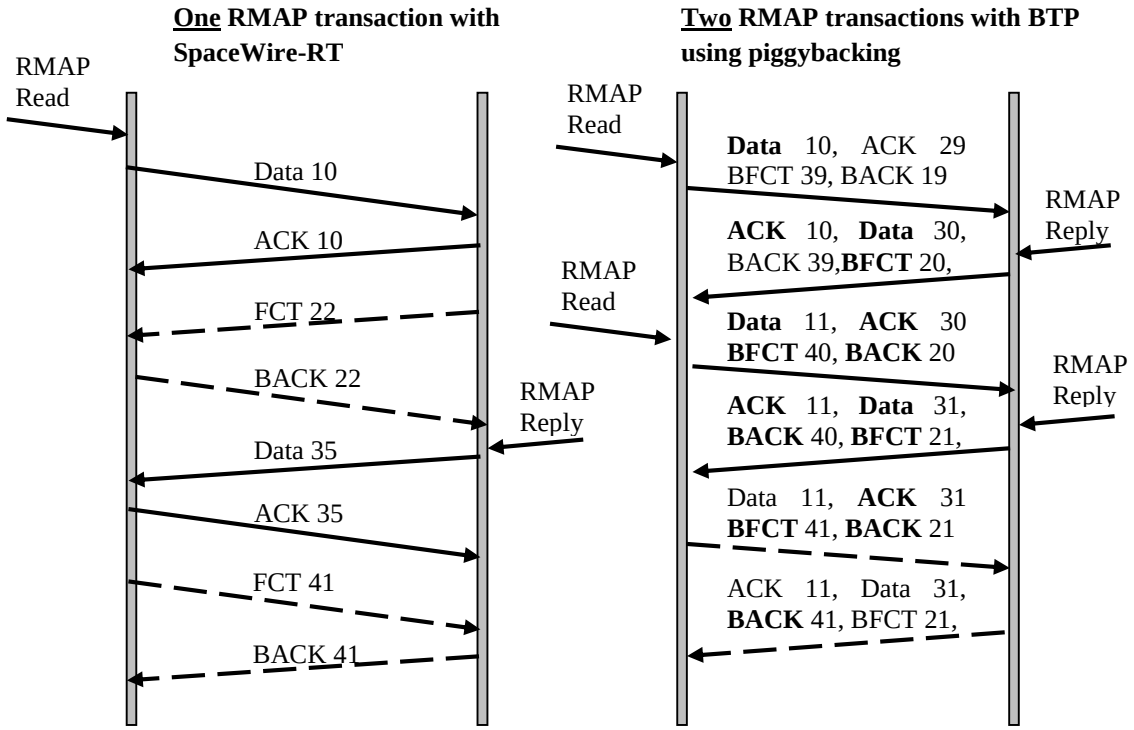


Figure 5-1: Piggybacking reduces the packets required for a RMAP transaction

The implementation of piggybacking seems straightforward but care must be taken when trying to put together the acknowledgement and the end-to-end flow control notification in the same packet header. They are generated at different times:

- The acknowledgement is generated when the data packet is received.
- The end-to-end flow control is generated when the data packet is read by the user application.

Therefore, with asynchronous systems a deadlock situation could occur when:

- The sender does not send a PDU with data because the destination buffer is full.
- The receiver side does not send a PDU when the destination buffer is not full because it has no data to send and the acknowledgement of the last data received was already sent.

There are two possible complementary solutions:

1. Implement a timeout mechanism which starts when there is no buffer space or when a SDU or new flow control information is sent. This timer is cancelled when an acknowledgement is received. When the timeout elapses a PDU without data is sent which triggers the sending of updated acknowledgement and end-to-end flow control.
2. The receiver sends a PDU with new end-to-end flow control information even if there are no pending SDUs or acknowledgments to be sent.

The first solution is implemented by the TCP protocol, which also uses piggybacking, because this mechanism it is also required to cover the case of PDUs with acknowledgements being lost. However, the second mechanism gives a faster reaction time, so BTP has implemented both solutions.

5.1.2.2 Segment Size

The maximum segment size (SDU size), is the size of the largest portion of a user message that can be encapsulated in a single PDU or SpaceWire packet. Table 5-2 shows the effective maximum raw data rate at 200Mbit/s link speed for different SpaceWire devices and 255 bytes of segment size. Note that the transmission time of 255 bytes is only 12.75 µs and the maximum possible raw data rate of SpaceWire is 200Mbit/s. For the net data

rate term, which accounts for SpaceWire protocol overhead, the maximum possible data rate is 160Mbit/s.

Table 5-2: Maximum raw data rate at 200Mbit/s with 255 segment size

Device type	Inter-packet delay	Raw data rate
Hardware packet generator (STAR-Dundee packet generator)	< 1 us	> 177Mbit/s
Space qualified embedded CPU (RTC)	15 us	88 Mbit/s
Windows PC (STAR-Dundee BRICK)	800 us	3 Mbit/s

Note that in BTP, only the last segment of a message is smaller than the maximum segment size and a PDU cannot contain two segments corresponding to different user messages.

SpaceWire-RT limits the segment size to 255 bytes which can reduce the performance depending on the inter-packet delay or the router switching delay. However, small segments reduce the worst case packet delivery time in non-synchronous networks. Small segments also allow simpler hardware implementations with small receiver buffers or to have more channels with the same available memory. Typically the receiver buffer can store multiple segments in order to keep receiving data while the acknowledgments and flow control tokens are being sent, so the maximum data rate can be achieved.

In synchronous networks, the segment size must adapt to the timeslot period if only one segment is allowed to be sent per timeslot. The minimum timeslot period is limited by the Time-Code latency, which is half of a microsecond per hop when using the SpW-10X router.

BTP leaves the maximum segment size as a parameter. The value of the SDU size field in the packet format indicates the maximum segment size. This allows the receiver to compute the receiver buffer size required to store multiple maximum-sized segments and achieve the highest performance. The maximum value is 16 Kbytes.

5.1.2.3 Channel Initialisation

SpaceWire-RT did not specify how the channel was initialised. It was assumed that other protocols were used to reset the state parameters of the protocol. This is a clear limitation, since it relies on the reliability of other protocols which may not follow a schedule when a synchronous network is considered.

Connection-oriented protocols exchange protocol capabilities when the connection starts and they are usually implemented in software. However, BTP protocol aims to be as simple as possible, so the simplest possible mechanism was used. In fact the only configuration required for the initialisation of a channel derives from the need to reset all state information, basically the sequence numbers. This can be done with a single bit, implemented as a flag in every packet header. Another bit is used to acknowledge the reception of a reset signal, in order to support reliable connection setup. During protocol initialisation the source sends a PDU with the reset flag set and waits until a PDU with the acknowledge flag set is received.

This simple mechanism also enables recovery from critical node errors. A redundant sending unit can reset an active channel in a node that was receiving data from a sending node that failed. The data transfer can then be resumed from the redundant unit to the same destination using the same channel number. Figure 5-2 shows the sequence of events described.

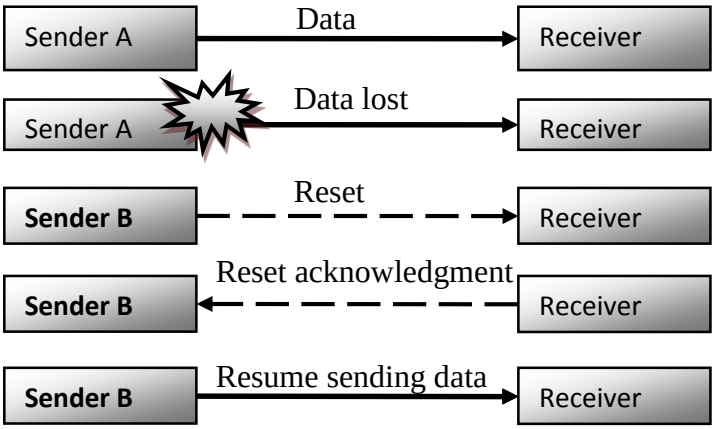


Figure 5-2: Recovering procedure using a redundant unit

5.1.2.4 Channels Identifier

Another difference with SpaceWire-RT is the decoupling between channel number and the destination node. In BTP the channel number is node-unique instead of network-unique. This allows having more than 256 different channels or data flows in a network, plus the possibility to assign a channel number depending on the application type of the communication.

FIFOs Node A

Channel	*Tx buffer*	*Rx buffer*
1	1	1
2	2	2
3	3	0
4	0	3

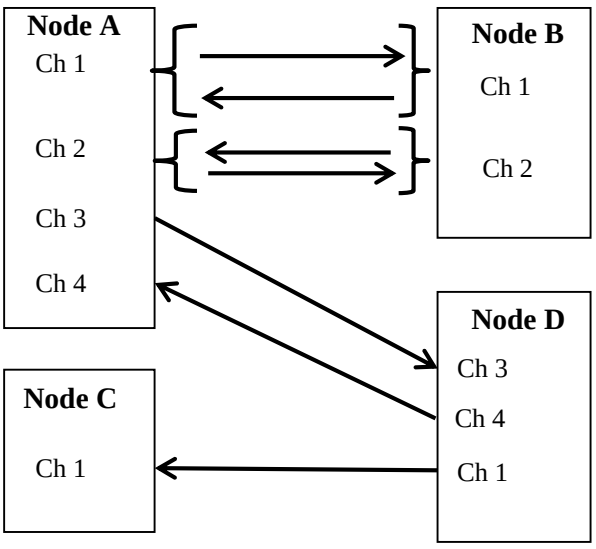


Figure 5-3: Example of protocol channel assignments

Figure 5-3 shows the flexibility of the channel identification scheme proposed. BTP allows unidirectional or bidirectional channels. The actual receiving and transmitting buffers can be assigned independently for each channel number or channel ID. However, channel IDs between a pair of nodes must match. This impose some limitations in the channel number allocation but allows a simple packet header processing, as the actual transmission and reception buffer does not need to be specified. A look up table is used to translate the channel ID to the Tx and Rx buffer ID, as shown in Figure 5-3 for the Node A.

5.1.2.5 Return Path Address

Given a source-destination pair, the return path address is the SpaceWire path address used to send the acknowledgements and flow control information. As explained, SpaceWire network routing can use logical or path addressing. The path redundancy feature of the SpaceWire-RT protocol requires configuration in each node of a list of destination and return paths address for each channel, prior using that protocol. Table 5-3 shows the format of a channel configuration example. The path index points to a path table with the destination and return paths, an example of which is shown Table 5-4.

Table 5-3: BTP channel configuration example

Channel	Destination logical address	Priority	Path index in the path table
1	120	High	1
2	120	Medium	1
3	121	Low	2

Table 5-4: BTP path table example

Path Index	Primary Path	Redundant Path	Return Path
0	reserved	reserved	reserved
1	1, 3	1, 4	1
2	2	2	3

In order to provide an optional mechanism that allows the use of the BTP protocol to send data to a remote unit that does not have any previous configuration, additional information needs to be provided within the PDU. The BTP protocol includes a one-byte field in the PDU packet format called return path. The "Return Path" column in Table 5-4 gives the value to set in the return path field of the BTP PDU.

The value of the return path field determines how the mechanism works:

A. When it is zero and the node does not have a channel configuration table, it means that SpaceWire logical addressing should be used with the source logical address provided in the BTP packet format.
B. When it is smaller than 32 but not zero, the value is used as an index to a path table previously configured in the node. The path table contains for each entry, the primary and redundant network path address used to send BTP PDUs.
C. When the return path value is larger than 32, it does not indicate a path index and instead this return path byte is used as the first byte of a reply command packet. The routing tables must be configured to route the packet back to the data source node using the return path byte and logical addressing. The last router in the return path, which is connected to the source node, should be configured to remove this byte.

In case there is no path table configured, only mechanisms A and C can be used. The advantage of using C instead of A is that it decouples the logical address used usually as a node identification from the actual network path. They are different concepts and it is useful to separate each function.

One or more redundant paths can be specified in the path table. They will be used when indicated by the redundancy field in the PDU received. When the data source node detects that it does not receive any acknowledgments it sends a PDU which indicates in its redundancy field which redundant path the receiver should use from the ones available in the path entry indicated by the return path field of the PDU. In case there is no path table, the return path value will be the return path field plus the value of the redundant path field. This mechanism provides enough flexibility to accommodate multiple use cases using intelligent, passive or not configurable nodes.

5.1.2.6 Scheduled Mode

The scheduled mode of SpaceWire-RT divides each timeslot in two parts:

- Data transfer phase: When the timeslot starts, zero or one data packet and zero or more Buffer Flow Control Token (BFCT) packets are sent.
- Acknowledgment phase: When a data or a flow control packet (BFCT) is received, the corresponding acknowledgment packet is sent.

This scheme has the problem that the timeslot duration needs to account for the sending of a segment and flow control data and the waiting time until the acknowledgments for both are received. This depends on the inter-packet delay introduced by the sender and the receiver which increases the latency and reduces the throughput.

BTP unifies all SpaceWire-RT PDU types into a single packet format. When sending a PDU with data at the beginning of a slot the acknowledgment and flow control information is also provided. Therefore the simplest approach is to allocate for a timeslot the time required to send just one maximum sized PDU. There is no need to allocate time for the acknowledgement as this can be received in the next slot.

Figure 5-4 shows the difference between the two approaches. The BTP can send the same amount of information with a smaller timeslot duration.

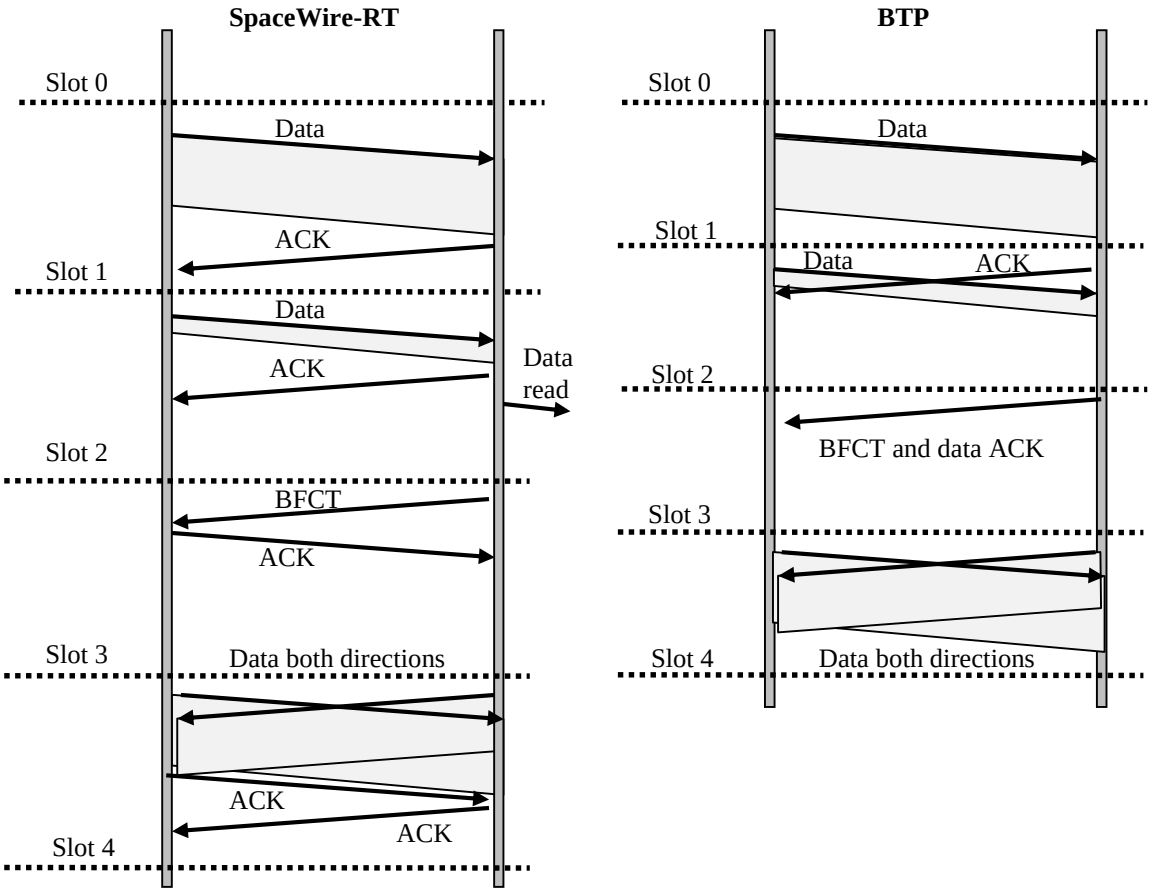


Figure 5-4: Timeslot duration for SpaceWire-RT and BTP

It is important to note that when using SpaceWire-RT in scheduled mode, the retry mechanism uses a send and wait scheme. In contrast, BTP uses a sending window in both asynchronous and scheduled mode. The acknowledgement is received in a subsequent

timeslot when there is no other priority channel with pending data or end-to-end flow control information to be sent.

There is still an issue with the handling of end-to-end flow control with the scheduled mode proposed. As only one PDU is sent per slot, the sending of end-to-end flow control channel information uses a complete slot even if there is no other information to send. This wastes a lot of bandwidth when using multiple channels per source-destination pair. The solution is to provide, in a single PDU, end-to-end flow control information of multiple channels corresponding to the same source-destination pair.

This solution is implemented without using relative values (i.e. BFCT values) or sequence numbers but using instead a simple XON/XOFF scheme that only requires a single bit per channel. When the receiver only has free buffer space for one or zero maximum sized segments the receiver clears (OFF) the corresponding XON/XOFF bit, otherwise the bit is set (ON). BTP uses one byte field for XON/XOFF which allows up to eight channels for each source-destination pair which is enough for most scenarios.

Figure 5-5 shows an example of this mechanism. When slot 0 starts, the receiver buffer of channel 1 in node B has space only for a single maximum sized segment so it sends a PDU with the XON/XOFF bit of this channel cleared (XOFF). In the same slot this buffer is completely filled by the PDU received. Later (slot 1), the buffer is read by the user application so there is space for more than one maximum-sized segment. In the next slot (slot 2) the receiver sends a PDU with the XON/XOFF bit of this channel set (XON). The sender continues sending data from channel 1 which has higher priority than channel 2. Note how this scheme allows encapsulation of end-to-end flow control information of channel 1 within the PDU that contains a data acknowledgement for channel 2, increasing the overall efficiency.

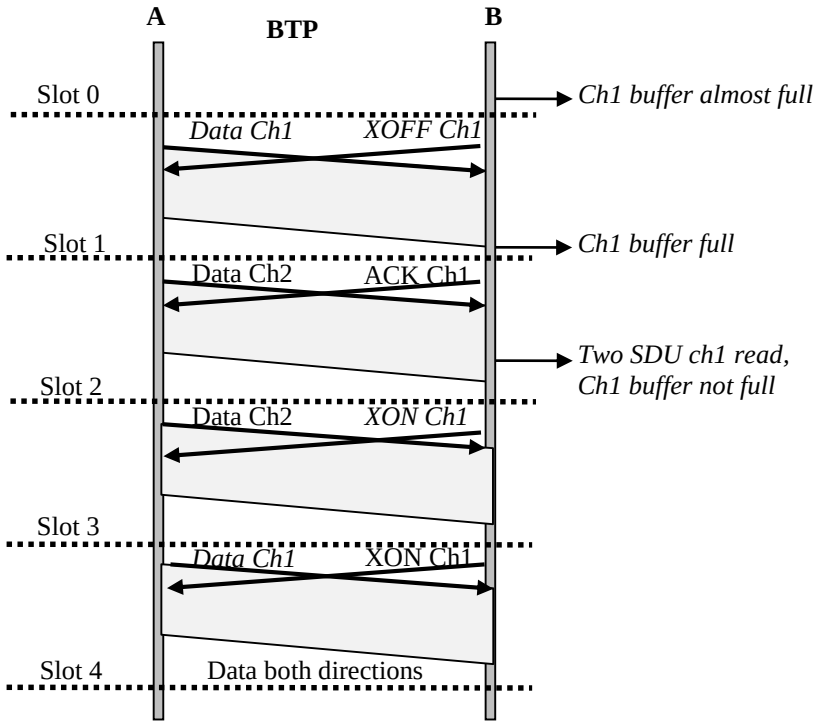


Figure 5-5: BTP XON/XOFF end-to-end flow control

5.1.2.7 Link Monitoring

Two extra fields are included in the BTP packet header. They provide the following information:

- Last Time-Code value received.
- Priority and QoS of the PDU.

Besides being useful for link monitoring purposes, they can also be used by the receiving node. The Time-Code value can detect outdated packets in a synchronous system and the priority field can be used to set the sending priority of the acknowledgement and flow control packet for this channel in case it is not previously configured.

5.1.3 Protocol Specification

The BTP protocol has two modes, the asynchronous mode and the scheduled mode. They all share a single common packet format or Protocol Data Unit (PDU). BTP performs the segmentation of a user message with an arbitrary size in multiple segments. Each segment defines the Service Data Unit (SDU) within a PDU. The PDU is the cargo of a SpaceWire packet which may have one or more leading bytes which form the SpaceWire destination path address.

5.1.3.1 PDU Format

Figure 5-6 shows the PDU format consisting of a header with multiple fields, the SDU, and the data CRC covering the SDU. Figure 5-7 shows how the PDU is encapsulated within a SpaceWire packet.

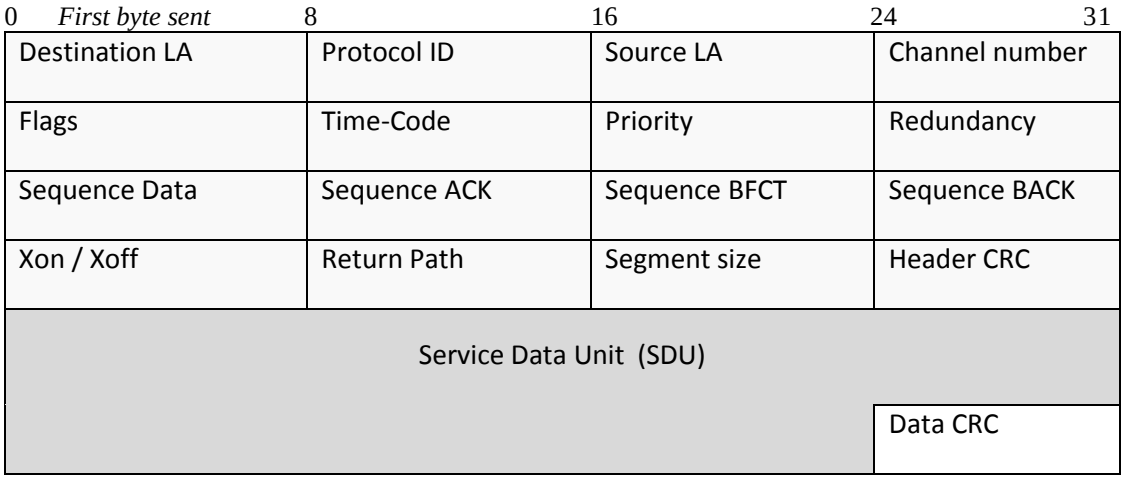


Figure 5-6: BTP Protocol Data Unit (PDU) format

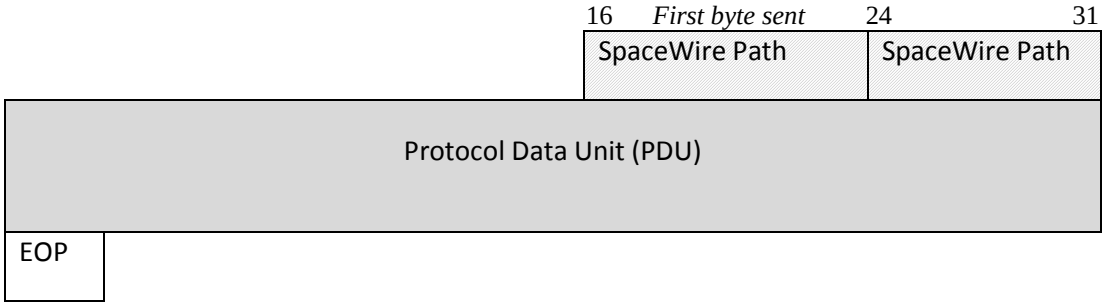


Figure 5-7: Encapsulation of a PDU within a SpaceWire packet

The fields of the BTP PDU are explained in Table 5-5

Table 5-5: BTP header fields

Field	Description
Destination LA	Destination Logical Address
Protocol ID	Protocol Identifier
Source LA	Source Logical Address
Channel number	Channel number used to identifies the destination buffer.
Flags	bit 7: Scheduled mode bit 6: Start of message bit 5: End of message bit 6: Reset bit 3: Reset signal acknowledgment bit 2: Data acknowledgement requested bit 1: Reserved bit 0: Reserved
Time-Code	Last Time-Code value received
Priority	Sending priority
Redundancy	bit 7-4: Number of paths bit 3-0: Current path used
Sequence Data	Data sequence. Incremented when a SDU is sent.
Sequence ACK	Data acknowledgement sequence. Incremented when a valid SDU is received
Sequence BFCT	End-to-end flow control sequence for non-synchronous mode. Incremented when a maximum sized segment is read by the received from the receive buffers.
Sequence BACK	End-to-end flow control acknowledgement sequence. It matches last valid BFCT sequence received.
XON / XOFF	Each bit indicates to the sender if the corresponding channel has space in the receiver buffer. Bit 0 is the first channel number of the source-destination pair.
Return Path	Provides the SpaceWire path address for the receiver to send an acknowledgement.
Segment size	Indicates the maximum segment size being used in multiples of 128 bytes.
Header CRC	Cyclic Redundancy Check covering the header.
Data CRC	Cyclic Redundancy Check covering the SDU.

5.1.3.2 Asynchronous Mode

In asynchronous mode, when a new SpaceWire packet with a PDU can be sent, the sending interface selects the channel with the highest priority and with new information to deliver, which happens when one or more of the following conditions occurs:

- A new SDU has to be sent or a pending SDU has to be resent due to an error being detected.
- An acknowledge has to be sent for a SDU or end-to-end flow control
- New status information of the end-to-end flow control has to be sent.

The receiver continuously accepts valid incoming BTP packets and processes each sequence field independently. The priority associated with the channel is updated with the respective field from the last received packet.

5.1.3.3 Scheduled Mode

In scheduled mode each channel has a list of valid timeslots that determine when the channel may send data. Timeslots are determined by the Time-Codes (TC) received. The timeslot duration equals to the Time-Code period.

The sending interface selects the channel with the highest priority and with new information to deliver that it is allowed to send during the next timeslot. The sender sends the SpaceWire packet with a PDU when a Time-Code is received.

Note that if a PDU is not received in a timeslot the current end-to-end flow control information (provided by the XON/XOFF field) is considered invalid until a new PDU arrives.

5.1.4 Experimental Apparatus

In order to evaluate the BTP protocol an experimental apparatus was developed in software for an EGSE unit. Specifically, the prototype runs over the Windows platform using the STAR-Dundee USB Brick unit which allows to send and receive SpaceWire packets with the available software drivers.

The prototype has two main components:

- The protocol kernel implemented as a software DLL component in C language.
- A Graphical User Interface (GUI) which allows configuration and monitoring of the status of the node using BTP. This BTP validation software makes software calls to the protocol kernel DLL.

This architecture has the advantage of achieving the maximum SpaceWire performance with the protocol while making it simple for the user to control the system using the GUI.

5.1.4.1 BTP Kernel Software

The BTP kernel implements the BTP protocol specifications in C language and exposes an API interface so the BTP protocol can be used by another application, in this case a GUI but could be also a script simulating a particular space application. Figure 5-8 shows a simplified architecture of the protocol kernel implementation. It is based on two threads that keep the data structures accessed by the external user application calls, updated with the latest BTP protocol status. The Receiver thread receives the incoming SpaceWire packets, processes them following the BTP protocol, and outputs the user messages received correctly. The Sender thread process the messages to be sent by the user and sends the BTP packets required to achieve the requested QoS using for example segmentation or scheduling mechanisms.

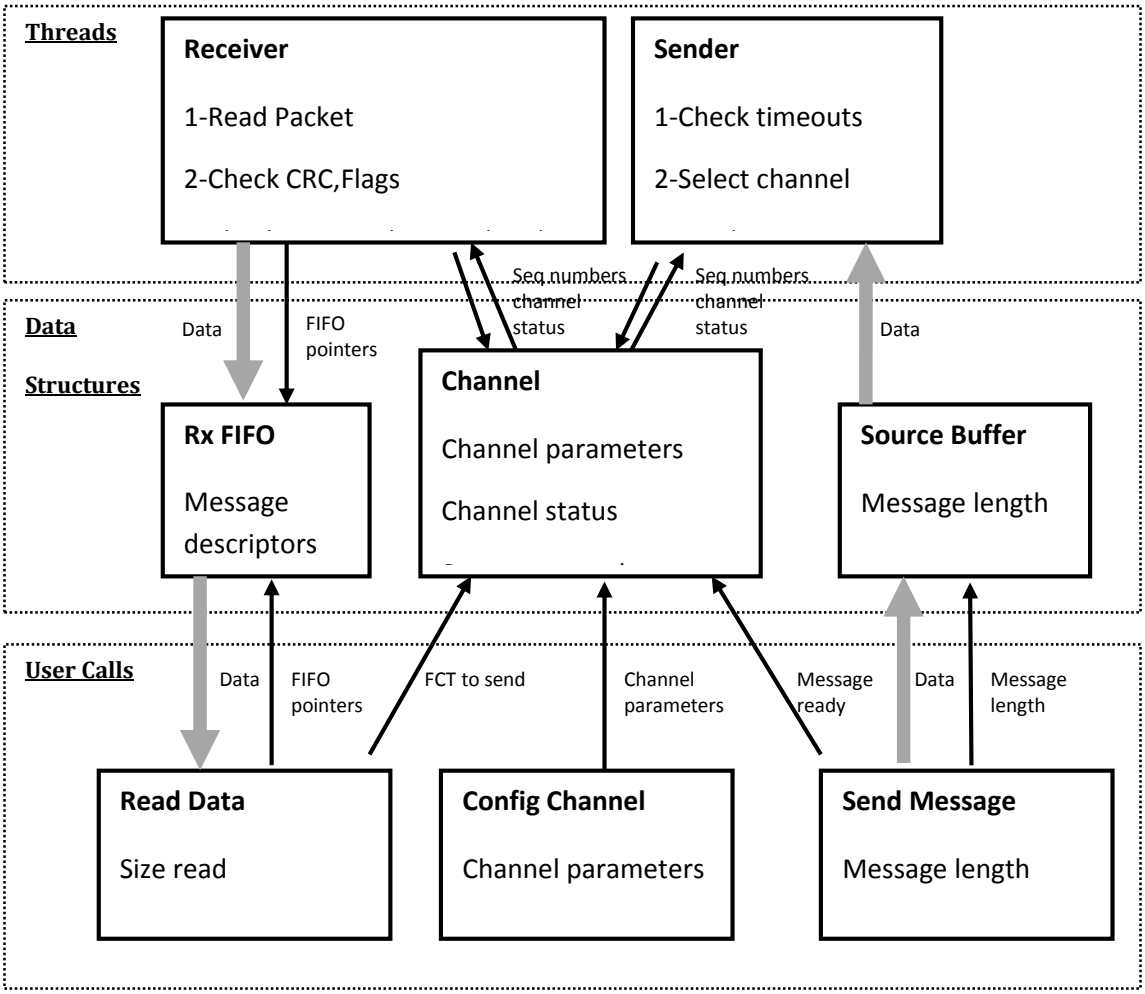


Figure 5-8: BTP Implementation architecture

5.1.4.2 BTP Validation Software

The SpaceWire BTP validation software is a user application that uses the low-level BTP library to send and receive files with a GUI interface. It supports all operational modes of the BTP protocol and can be used to show the protocol operation or as a validation tool.

The main features of the BTP validation software are:

- Graphical configuration of all BTP protocol capabilities, including:
    - Synchronous and Asynchronous mode.
    - Time-Codes sending function.

- Multiple channels can operate simultaneously.
- User can select multiple files to send at once from an arbitrary folder with the possibility to pause or cancel the transmission or the receiving of a file.
- Receiving and sender data rate sliders available for dynamic value setting.
- Receiving and sending statistics including errors, retries, PDUs, etc.
- Log window detailing the operation of the RT software library.
- User can select destination folder of received files.
- Link error and CRC error injection.

Figure 5-9 shows the channel configuration and the path table setup. Figure 5-10 shows a screenshot of the BTP validation software while operating with three channels with their status information updated in real-time.

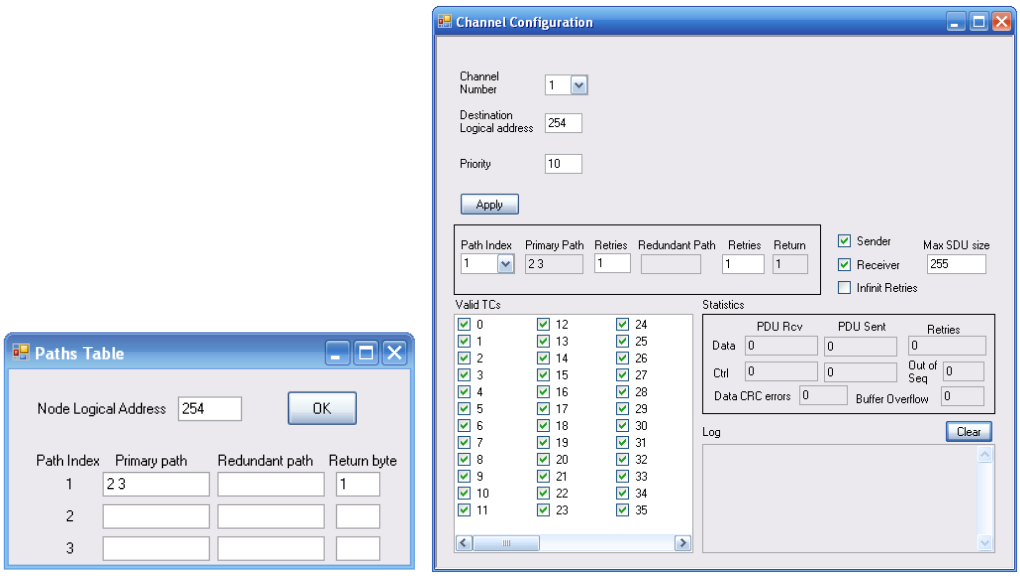


Figure 5-9: BTP configuration of path table (left) and channels (right)

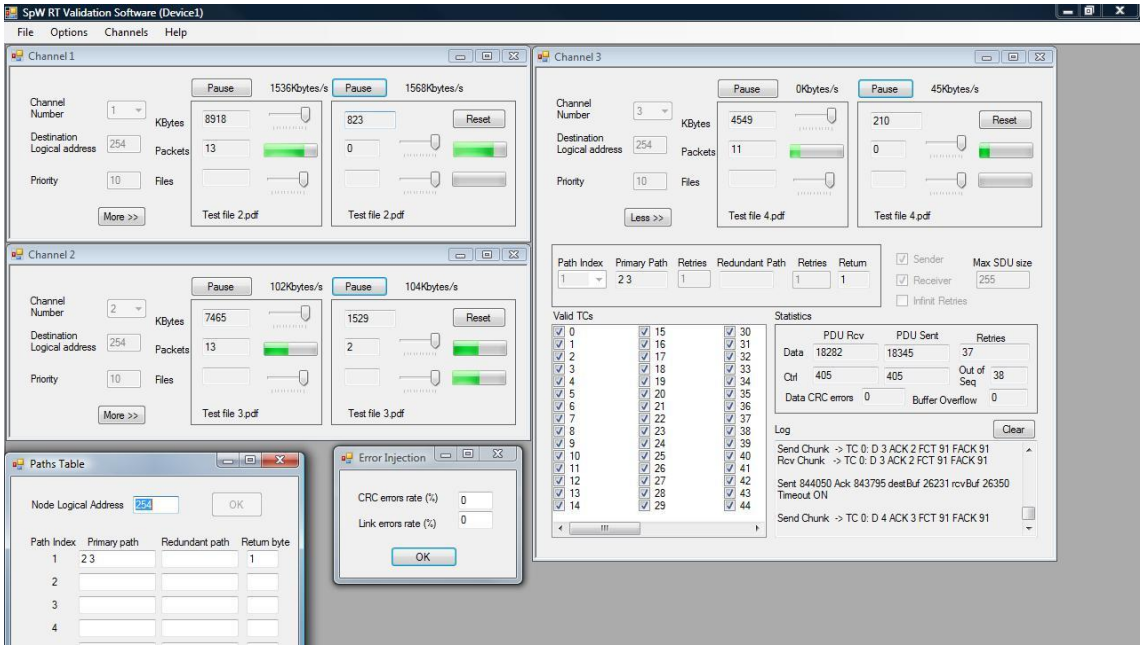


Figure 5-10: BSP validation software

5.1.5 Protocol Evaluation

The experimental apparatus described in the previous section was used to evaluate the performance of the BTP protocol compared with the results obtained without any transport protocol or using the initial SpW-RT protocol. For clarity, the evaluation is separated into the two main components of Quality of Service: reliability and timely delivery of messages. For each one, a test scenario is described first that it is as simple as possible so the experimental results can be compared with the ones obtained via an analytical study. This simple test scenario allows to validate the results and to better understand the benefits of using this protocol. The experiment is then run using each time different capabilities of the protocol so the improvements in the QoS network metrics for each protocol configuration can be seen.

5.1.5.1 Reliability

In section 4.1.4 it was shown that the retry mechanism of the SpW-RT protocol was an error detection and recovery technique that provided the required reliability. One of the objectives of BTP was to reduce the cost introduced by this capability in terms of protocol header overhead and number of packets sent per segment of user message. This cost implies a reduction of the maximum throughput that can be achieved by the SpaceWire protocol with a specific equipment. Therefore, the reduction of this cost by BTP should improve the throughput achieved. This is especially important for software implementations as there is a higher delay associated with the sending of a SpaceWire packet than with hardware implementations.

To measure that improvement the simplest network topology, a point-to-point connection, is used. This ensures that there is no delay related with the network congestion so the maximum throughput can be measured. The throughput is measured by counting the number of messages received during a large period of time. The user message size is fixed in this experiment to 32 Kbytes. The measurements are performed multiple times using different segment sizes and under two main configurations: bidirectional and unidirectional data transfers. In the bidirectional configuration the two experimental apparatus are sending messages continuously, while in the unidirectional only one is sending data.

The experimental setup is executed using the SpW-RT prototype described in section 4.1.4 using the same hardware to send SpaceWire packets, the STAR-Dundee USB Brick [STAR-Dundee 2010]. Finally the experiment is also executed without any protocol, i.e. unreliable raw SpaceWire, so each user message of 32 Kbytes is sent using a single packet. Note that SpW-RT segment size is fixed to 256 bytes and it is not customised as

in SpW-BTP. The Link Speed is set to a minimum value (2 Mbit/s) so the effect of the inter-packet delay or packet processing delay due to the Windows platform is minimised. Figure 5-11 compares the results.

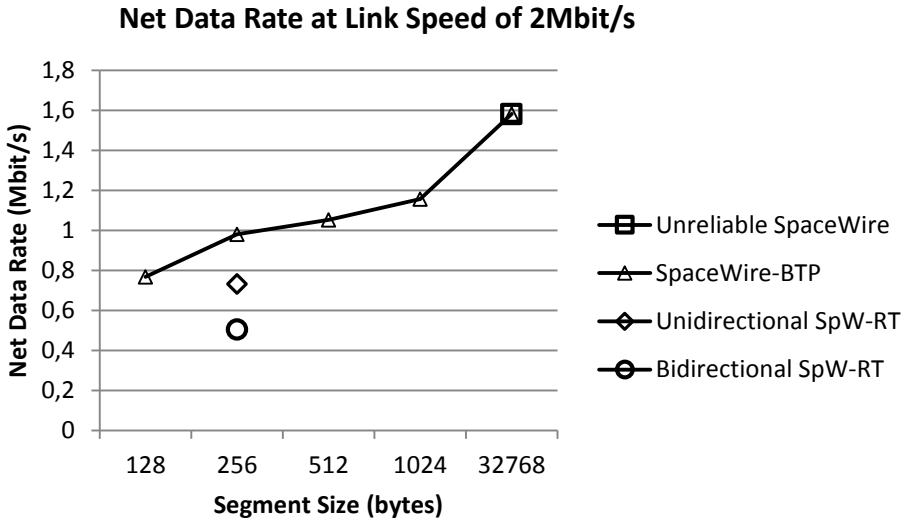


Figure 5-11: Net Data Rate measured at Link Speed of 2Mbit/s for each protocol

Note that only for SpW-RT, the net data rates measured for bidirectional and unidirectional data transfers are different. For SpaceWire-BTP and raw SpaceWire the results for both cases differ by less than 4% so they are not separately shown in the graph.

These experimental values can be easily explained taking into account the packet processing delay, i.e. the time taken by the USB Brick SpaceWire drivers between the indication to send a packet and the actual transmission of the first byte of a packet. The maximum throughput or data rate can be computed using equation (5.1):

$$D = \frac{\sum P_{size}}{\sum T} = \frac{\sum P_{size}}{\sum \left[T_{tx} + T_{delay}\right]} = \frac{\sum P_{size}}{\sum \left[{}^{P_{size}}/_{S} + T_{delay}\right]} \qquad (5.1)$$

where D is the data rate, P_{size} is the packet size, S is the link speed and T_{delay} is the packet processing delay. For simplicity, it is assumed that this delay is independent of the size of the packet. The summation indicates the need to sum all packet sizes and the time

it takes to send these packets, corresponding to a single user data segment. The packet size is computed taking into account the overhead of each protocol using the values of Table 5-6.

Table 5-6: Protocol overheads

Header	Bytes
SpW-RT Data Packet header	10
SpW-RT Control Packet header	8
SpW-BTP Packet Header	16

Protocol	#Control Packets
Unidirectional SpW-RT	1
Bidirectional SpW-RT	3
SpW-BTP (Unidirectional & Bidirectional)	0

The efficiency of the protocol is computed as the ratio between the user data rate and the link speed. Note that in this thesis when computing the overheads of each protocol developed, the 80% protocol efficiency of raw SpaceWire is not included (i.e. SpaceWire has 1.6Mbit/s net data rate at 2Mbit/s link speed). Therefore, in this analysis, one byte of user data requires the transmission of 10 bits. Figure 5-12 shows the values computed using a T_{delay} of 1.27 milliseconds which was the average value experimentally obtained.

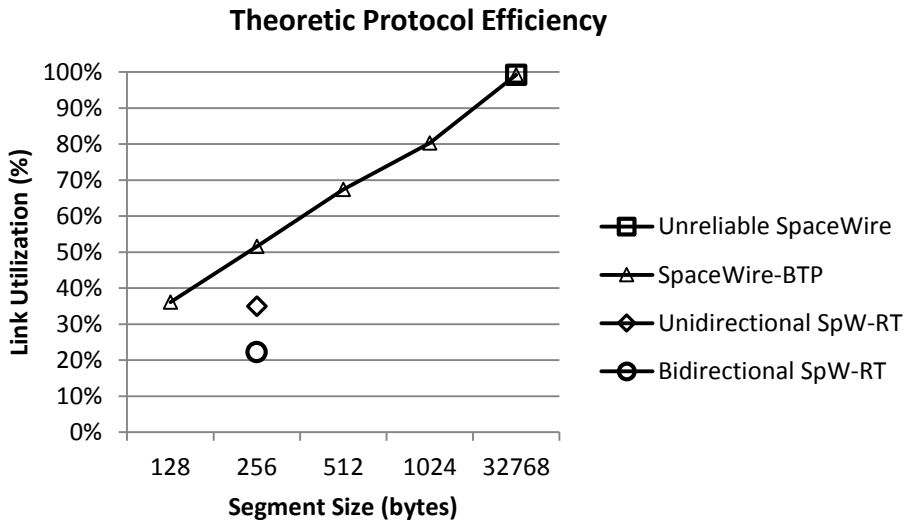


Figure 5-12: Protocol efficiency

This figure is very similar to Figure 5-11. The fact that the actual T_{delay} has some dependency on the packet size makes up most of the differences.

These results demonstrate the higher efficiency of BTP protocol versus the original specification of SpW-RT, thanks to the use of a single packet per segment of a message and the customization of the segment size. When the segment size is equal to the user message size, the overhead of the protocol is minimal. Therefore it has been proved that a reliable communication can be provided with minimum cost.

5.1.5.2 Timely Delivery

The aim of this section is to measure the capabilities of the SpW-BTP protocol in reducing the worst case delivery time of short control messages. The average delay was already analysed using simulations in section 4.2.8.

5.1.5.2.1 Priorities and Segmentation

For this evaluation the gain obtained using different QoS mechanisms of BTP is compared against not using any QoS transport protocol. The topology shown in Figure 5-13 is used, where each node represents an experimental apparatus.

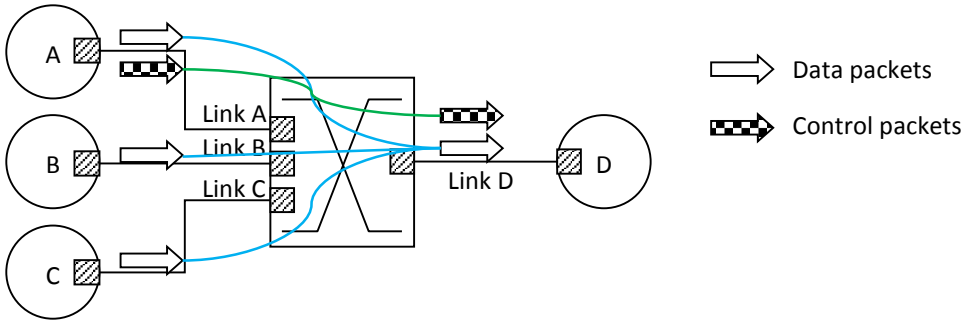


Figure 5-13: Topology for measuring timely delivery

All nodes are set up to send data packets of 8000 bytes to node D with the link utilisation specified in Table 5-7. The link speed is set to only 2 Mbps in order to minimise the impact of the variation in the packet processing delay due to the limitations of a PC software implementation of BTP. This allows a high link utilisation when sending large packets continuously.

Table 5-7: Link utilisation used for timely delivery evaluation

Link	Link utilisation
A	40%
B	20%
C	20%
D	80%

After some time each node has started sending data packets, node A is set up to send periodically a control message of 128 bytes to node D, which is sent in a single SpaceWire packet. A new control message is not created if there is a pending control message to be sent, so there is no need for a control message queue. The latency of each control packet is measured by adding a timestamp when the packet is created which it is checked at the receiver. The latency obtained is subtracted by the one obtained in a point-to-point connection so it is left only the delay due to the network, basically the network congestion. After the experiment has been run for a significant time, the highest control packet latency registered is recorded.

Table 5-8 shows the results for different segmentation and priority settings for the experimental apparatus running in asynchronous mode.

Table 5-8: Maximum delay measured due to network congestion

Maximum control packet delay	**Without sending priority** (single queue for data and control)	**With sending priority** (different queue for data and control)
Without data segmentation	355ms	112ms
Data segments of 512 bytes	18ms	11ms
Data segments of 256 bytes	9ms	7ms

The data packets are created with an exponential distribution, so without using priorities, the theoretical maximum control packet latency is infinite, but with higher values being less likely to happen and being recorded. Note that in case of a constant inter-arrival time for data packets the latency without using priorities would be similar to the one using priorities and different queues, because the common queue never holds more than one data packet.

The maximum delay of control packets introduced by the network when using priorities can be easily estimated using the following worst case:

1. The control packet is generated in node A of Figure 5-13 when nodes A,B,C have just start sending a data packet and the first link selected by the router is node A.

Equation (5.2) computes the estimated maximum delay D in the scenario described using the packet length L (in raw bits) and the link speed S. This simple equation assumes that the inter-arrival time of control packets is higher than the maximum delay computed.

$$D = N_{links} * \frac{L}{S} = 3 * \frac{8000 * 10}{2000000} = 120ms \qquad (5.2)$$

The value obtained is not exactly the one obtained experimentally (112ms) due to the worst case calculated being very unlikely.

The segmentation mechanism allows improvement of this result by modifying the parameter L. A short control message can fit in a segment size, which in SpW-BTP can be configured and be different for each sending entity.

Figure 5-14 shows how the delay depends on the segment size using equation (5.2) when the number of competing data packet flows (N_{links}) is one, two, or three (which is the experimental scenario presented). Note that for small segment values it differs from the experimental results (Table 5-8 with segmentation and sending priorities) due to the limitations imposed when sending SpaceWire packets using a software implementation.

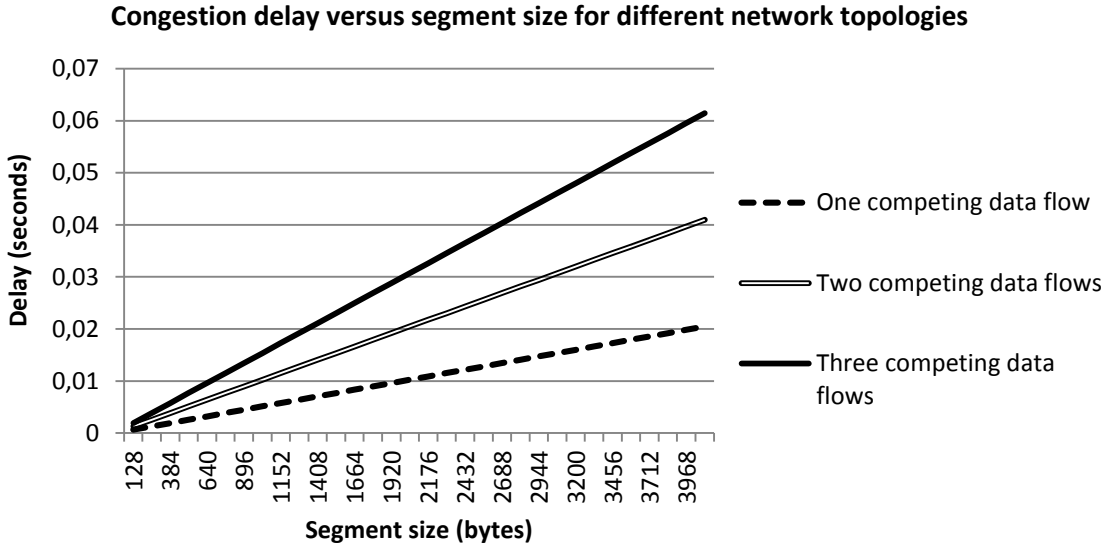


Figure 5-14: Network congestion delay of control packets versus segment size

5.1.5.2.2 Scheduling

Finally, the scheduling capabilities of BTP are evaluated. The aim is to reduce the maximum latency of the control packet.

The first parameter which needs to be determined when configuring a scheduled network is the timeslot period, which should be the same for all nodes that are interconnected. With BTP the timeslot period should be large enough to allocate the largest segment size,

or message size if segmentation is not used, plus the 16 bytes of the BTP header. Then it should be added the maximum variation in time that it takes for a node to start sending a packet. This is usually very small for a hardware implementation but in this case the BTP protocol has been prototyped in a software running in a PC, for which a delay of up to 10ms has been measured. In the case considered, the message size is 8000 bytes which takes 40ms to transmit at 2Mbit/s. If segmentation is not used, the packet size is almost equal to the message size as the BTP header overhead is irrelevant. So, a timeslot period of 50ms can be set, which gives an 80% timeslot efficiency (see Figure 5-15). This means that up to 80% of the link capacity can be used using scheduling. This is just the highest link utilisation of the network (link D, see Table 5-7) so it can be ensured that the system is stable.

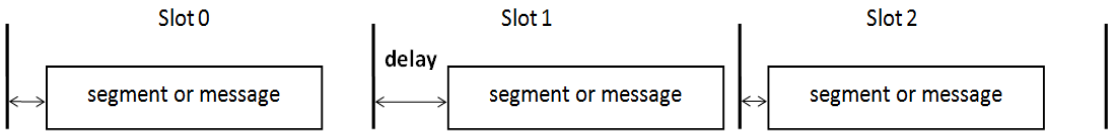


Figure 5-15: Transmission packet delay or inter-packet delay

The network scheduling table for the specific case described before (see Figure 5-13) is shown in Table 5-9, where N is any natural number. For each link, the packet flows that are allowed to send packets in each slot are given in the format "Source → Destination". Note that the source-destination pair "A→D" represents the allocation of two channels to this pair of nodes, the data channel and the control channel. As explained, the control channel has higher priority than the data channel so if there is a control packet pending to be sent it will be sent before a data packet.

Table 5-9: Network Scheduling table for BTP evaluation

	Slot 4*N+**0**	Slot 4*N +**1**	Slot 4*N +**2**	Slot 4*N +**3**
Link A	**A → D**		**A → D**	
Link B		**B → D**		
Link C				**C → D**
Link D	**A → D**	**B → D**	**A → D**	**C → D**

Using BTP in scheduling mode with this configuration, the measured maximum control packet delay (from node A to node D) was 102ms which is close to the expected value of 100ms (2 slots of 50ms). The expected value can be obtained with equation (5.3), where N is the maximum number of slots not assigned to the control packet flow between two slots that are assigned to this flow.

$$D = (N + 1) * T_{slot} = (1 + 1) * 50 = \ 100\, ms \qquad (5.3)$$

This simple equation computes the latency for the worst case, which occurs when a control packet is generated just after the beginning of a slot (4*N+0) and it must wait until it is sent at the beginning of next allocated slot (4*N+2).

The scheduling mode of BTP has reduced the maximum latency of control packets without using segmentation from 120ms to 100ms but it can also be used to increase the overall throughput of the network. For example, with scheduling additional packet flows can be added between nodes A,B,C filling all unused slots in the scheduling. They can be used to exchange bidirectional information between nodes A,B,C.

Table 5-10 shows the new scheduling table and Figure 5-16 shows the network status at each slot.

Table 5-10: Network Scheduling table for BTP evaluation using all slots

	Slot 4*N+**0**	Slot 4*N +**1**	Slot 4*N +**2**	Slot 4*N +**3**
Link A	**A → D**	**C↔A**	**A → D**	**B ↔ A**
Link B	**C ↔ B**	**B → D**	**C ↔ B**	**B ↔ A**
Link C	**C ↔ B**	**C↔A**	**C ↔ B**	**C → D**
Link D	**A → D**	**B → D**	**A → D**	**C → D**

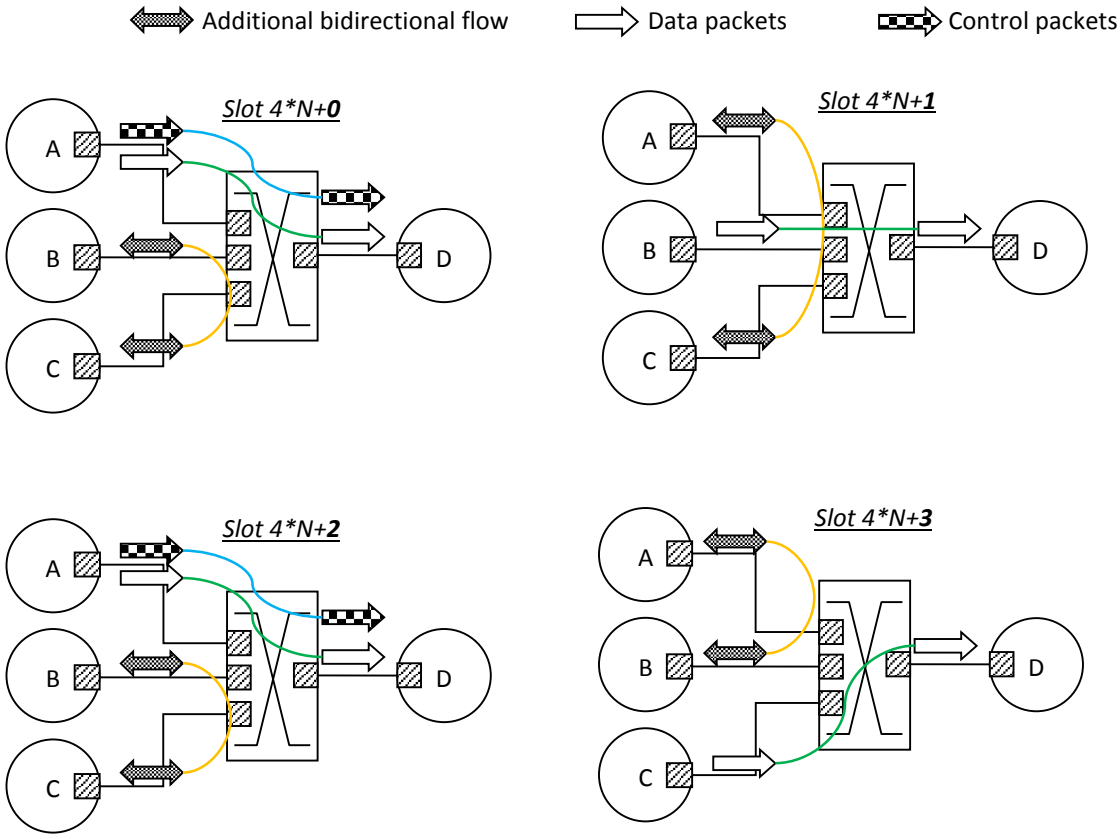


Figure 5-16: Packet flows for each slot

The most interesting fact of this new schedule table is that it provides an aggregated network throughput that it is not possible to achieve in an asynchronous network. If the network is not scheduled, congestion appears when multiple flows tries to access the same link. When congestion occurs the bandwidth of the link is unused so depending on the

source data rate of the flows it can lead to an unstable network, with source buffer overflows that produce data loss. Therefore under some scenarios, a scheduled network provides more aggregated throughput that an asynchronous one.

A simplistic analogy can be made comparing an asynchronous network with a roundabout and a synchronous network with using traffic lights. Depending on the traffic, one is more efficient than the other.

5.1.6 Summary

This section has evaluated how a generic QoS transport layer can be designed. BTP is a new transport protocol that provides the two main QoS aspects required, reliability and timely delivery, for generic asynchronous or synchronous network. It implements acknowledgements, priorities, segmentation and scheduling and the results from the experimental apparatus have proved that it can achieve higher throughput and lower latencies than when not using any transport protocol.

The BTP protocol is very well suited for bidirectional data flows but it is slightly inefficient for scheduled networks. When the data flow is unidirectional, the acknowledgement and the flow control information must be allocated the same bandwidth required as when sending a PDU with a complete SDU. This almost doubles the bandwidth needed when unidirectional data flows are considered. Unfortunately that use case is quite common in space applications, as there is an asymmetry between the instruments and the payload processing unit. The instruments sends at high data rate, while the payload processing unit sends basically small commands. In the next section a new protocol is presented that addresses these limitations.

5.2 Unidirectional Transport Protocol

The Unidirectional Transport Protocol (UTP) aims at providing a transport layer optimised for synchronous networks that carry highly sensitive command and control information, together with high data-rate payload data coming from the instruments.

It was designed considering the results of the BTP protocol and the feedback of the SpaceWire Working Group from the initial specification of SpaceWire-RT. This provided a new set of requirements related to synchronous networks, on which UTP is based. The outcome of this work is a new version of SpW-RT specification [Parkes 2009b], that it is called here UTP protocol. It was written by the University of Dundee in collaboration with the author of this thesis, who also did the experimental evaluation using space-qualified devices.

5.2.1 Requirements

The key idea was to design a protocol optimised for synchronous or scheduled networks consisting of a set of unidirectional data flows, with guaranteed latency and throughput. In addition to the requirements of section 5.1.1.2 the new protocol had to provide:

- Better QoS metrics (latency and throughput) than BTP for asymmetric data flows, by optimizing the timeslot allocation.
- Better QoS metrics than BTP for command and control operations.
- Smaller amount of memory required by the receiver buffers.

5.2.2 Design Considerations

One important consideration for the trade-off study of UTP is the use of unidirectional channels instead of the bidirectional channels used by BTP. This implies that it is not possible to apply the piggybacking technique, so it is required to have different packet types, similar to the ones defined in the original SpaceWire-RT definition [Parkes 2008d]. Instead of the unique packet header of BTP, in UTP there are data packets containing segments of user data and control packets that provide acknowledgements and end-to-end flow control information.

Another consideration is that UTP does not need to support asynchronous networks and it is not required to be able to achieve the highest protocol efficiency possible. Instead, achieving a higher throughput relies on a more efficient slot allocation for unidirectional flows. Also, the main application of UTP is command and control applications where a low latency is most important. A more efficient delivery of control packets will be achieved using a different timeslot scheme than BTP.

5.2.2.1 Timeslot Phases

Each timeslot can be divided in two or more phases, for the separation of user data and the protocol control information. The original SpaceWire RT divided each timeslot into data transfer and acknowledgement phases. This has the advantage of providing the acknowledgement in the same timeslot, which gives more robustness to command and control applications.

However, SpaceWire-RT did not handle very well the delivery of end-to-end flow control information, especially when dealing with multiple channels. A better scheme could be used that follows the three natural steps of a unidirectional data transfer:

1. The destination informs the data source node which channels have space in their receive buffers.
2. The source node sends multiple data packets of different channels in order of priority until all data is sent or the maximum number of data packets allowed in a timeslot is reached. It is also possible that some or all data packets belong to the same channel.
3. The destination sends the acknowledgements of the data packets received.

If a timeslot follows these three phases it minimises the amount of memory space required in the source and destination buffer. Both need only to have space for as many data packets as can be sent in a single slot. This is in contrast with BTP, which uses a sending window for both data and end-to-end flow control, and it requires space for data being sent during multiple timeslots. Therefore UTP will split the timeslot duration in these three phases, as shown in Figure 5-17. Note the acknowledgements (ACKs) and flow control information (BFCTs) are sent in the opposite direction to data packets (DPs).

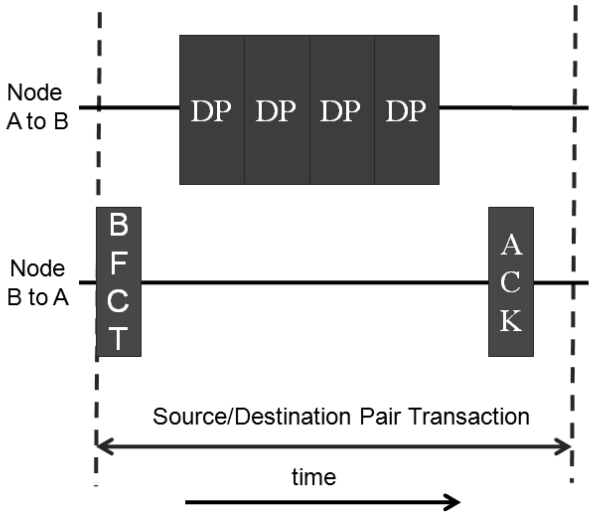


Figure 5-17: Source-destination pair transaction in a timeslot.

5.2.2.2 UTP Control Packets

It is also important how acknowledgements and end-to-end flow control information is encapsulated in control PDUs. Two basic rules improve the performance:

- Encapsulate control information regarding multiple channels in a single PDU. Therefore, one PDU is used for acknowledgement and one for end-to-end flow control. This reduces the overhead due to inter-packet delay.
- As end-to-end flow control information is received periodically at the beginning of a timeslot, the absolute free space value of each buffer is sent, instead of relative or accumulative values. With this scheme it is not required to acknowledge the reception of control PDUs that contains end-to-end flow control.

  If such a control PDU is lost and not received at the beginning of a timeslot, the data transfer phase does not occur until a new one is received in another timeslot. Note that with the relative values used by asynchronous BTP, a retry operation was required when end-to-end information was not acknowledged.

5.2.2.3 Timeslot Duration

The main QoS metrics of UTP, the latency and throughput achieved, are directly related to the duration of each timeslot, because the UTP aims to specify these metrics deterministically when the scheduling of UTP is designed.

The timeslot duration has to be the same for all nodes in the network considered. For the trade-off of the optimum timeslot duration the following needs to be considered:

- When the timeslot duration increases, the maximum protocol efficiency increases.

- When the timeslot duration decreases, the latency decreases and the schedule can be more optimised to a specific traffic scenario. The buffer memory required to hold the packets to be sent is also reduced.

The minimum duration of a timeslot is the time required to send one data packet plus the sum of the time required for the following mandatory operations:

1. The reception of a time-code with enough margin to accommodate the maximum jitter introduced by the network since it was sent by the time-code master.
2. The sending and reception of the end-to-end flow control packet. This time depends on the processing time, the control packet size and the worst case network latency when there is no congestion.
3. The sending and reception of the acknowledgement control packet.

Table 5-11 provides approximate timings for each of the previous operations assuming the control packets specified in section 5.2.3.1, a link speed of 200Mbps and a maximum distance of four hops between any node pair.

Table 5-11: Estimated timings for each timeslot phase of UTP

Timeslot phase	**Estimate duration**
Time-code reception (T_{TC})	1.2 us
Acknowledgement packet reception (T_{ACK})	0.9 us
End-to-end flow control packet reception (T_{FC})	5.4 us

Therefore the minimum duration of a timeslot is determined by the following equation:

$$T_{TS} = T_{TC} + T_{ACK} + T_{FC} + N * T_{DP} \quad (5.4)$$

Where T_{DP} is the time it takes to send one maximum sized segment (DP or data packet) of a channel and N is the maximum number of data packets that can be sent in the same

timeslot. T_{DP} depends on the maximum size of the SDU within a data packet. The timeslot duration and the Quality of Service achieved depends on these two parameters.

For the trade-off it is considered that the SDU size has to be a power of two and suitable values are 128, 256 and 512 bytes in order to limit the receiver buffer space required. For the number of data packets there should be at least three so at least three different channels can be used in the same timeslot. The maximum number of data packets is limited to 16 by the specification of the control packet format for end-to-end flow control.

Given a specific timeslot duration, or for each timeslot increment step, a possible optimum value for the SDU size and number of data packets is one that fills the timeslot with maximum efficiency using a maximum of 16 data packets.

Table 5-12 shows the possible configurations for a hardware and a software implementation using equation (5.4). The software implementation assumes an additional overhead per timeslot of 30µs due to the packet handling. This is an optimistic value but still in line with the value obtained experimentally in section 0.

Table 5-12: SDU size and number of DPs for increasing timeslot duration

Timeslot period (us)	**Hardware implementation**			**Software implementation**		
	SDU size	**#DPs (N)**	**Efficiency**	**SDU size**	**#DPs (N)**	**Efficiency**
50	256	3	77%	128	1	13%
75	128	9	77%	128	5	43%
100	128	12	77%	128	8	51%
125	256	8	82%	256	6	61%
150	**256**	**10**	**85%**	**256**	**8**	**68%**
175	**256**	**12**	**88%**	**256**	**10**	**73%**
200	**256**	**14**	**90%**	**256**	**12**	**77%**
225	512	8	91%	512	7	80%
250	512	9	92%	512	8	82%

The computations in Table 5-12 show that the use of a SDU size of 256 bytes is a good compromise between protocol efficiency, receive buffer size and low latency for both hardware and software implementation. Still, the timeslot can be adapted to a specific network scenario between 125µs and 200µs.

5.2.3 Protocol Specification

5.2.3.1 Packet Format

There are different packets formats defined. User data segments are encapsulated in Data Packets (DPs) with the header format specified in Figure 2-12. Acknowledgement packets (ACKs) are encapsulated as shown in Figure 5-18. Figure 5-19 shows the packet format of end-to-end flow control packets (BFCT) and Figure 5-20 shows the packet format of end-to-end flow control packets acknowledgments (BACK)

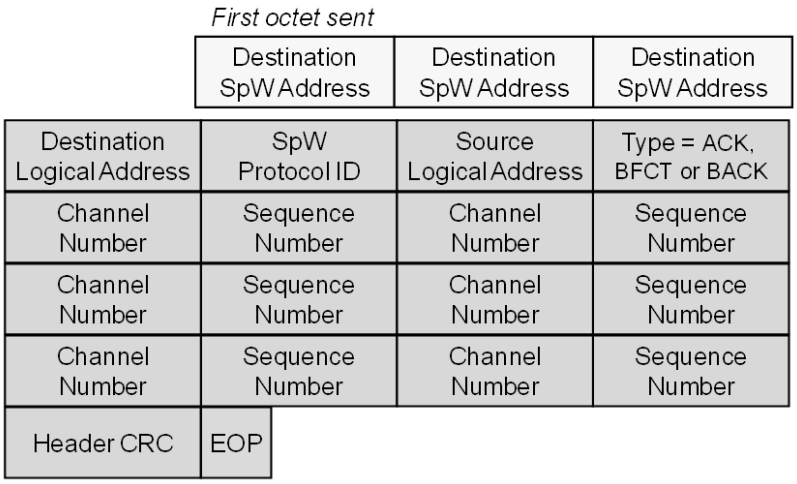


Figure 5-18: ACK packet format

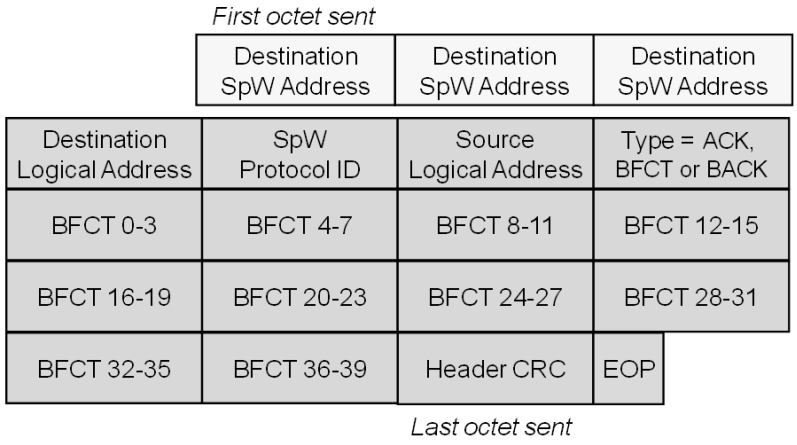


Figure 5-19: BFCT packet format

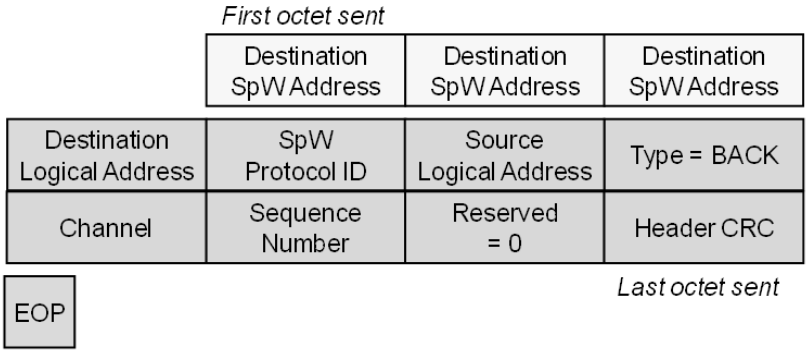


Figure 5-20: BACK packet format

5.2.3.2 Schedule Table

Timeliness of delivery is controlled by a schedule table used to specify which source channel can send information in which timeslot. Only one source is allowed to send information at a time, or multiple sources can send information at the same time provided that they do not use any common network resource i.e. send information over the same SpaceWire link. This provides deterministic delivery.

Scheduling can be combined with priority so that several channels, all to the same destination and using the same network resources for communication but with different priority levels, are mapped to a timeslot. Therefore the schedule table of every node has

for each timeslot a list of channels, ordered by priority, and a path identifier that defines a set of unidirectional links that constitute the route to the destination node.

Table 5-13 shows a simple example with two different timeslots where the sender is allowed to send data from multiple allocated channels to a destination node with logical address 200. Channel 1 is assigned the highest priority in both timeslots and will send a DP when it requests it, providing low latency for control messages. Channels 2 and 3 have been allocated from 50% to 100% of the bandwidth of these slots when control messages are not sent.

Table 5-13: Example of a schedule table

Slot	Logical Addr	Channels High » Low priority	Primary Path	Redundant Path
1	200	Ch 1, Ch 2, Ch 3	Path A	Path C
3	200	Ch 1, Ch 3, Ch 2	Path B	Path C

Channels 2 and 3 are used for sending data from two different sensors. If they are both generating data and there are no control messages each one will use half of the available bandwidth. If one sensor is disabled, the other one will have the full bandwidth.

There are two paths for each timeslot. The redundant path is only used when the primary path fails. However, different paths are used depending on the timeslot. This allows the system to operate with half of the bandwidth in case two paths fail, providing graceful degradation. However, channel 1 will not be affected as it has the highest priority.

Another aspect is that multiple DPs can be sent during a timeslot. Therefore, if a control message fits in a single DP the other DPs will contain data from the next lower priority channels until the maximum number of DPs are sent or no more data from all allocated

channels is available. Note that a lower priority channel is not used if there is still data to be send from a higher priority channel.

5.2.3.3 Timeslot Timing

Each timeslot has to handle a complete source-destination pair transaction including the transfer of flow control information (BFCTs) and acknowledgments (ACKs). The control information has to be scheduled as well, to avoid any congestion, taking into account that it travels in the opposite direction to the data (DPs). Therefore the timeslot timing is divided into three phases, as shown in Figure 5-17.

1. Buffer flow control phase: the receiver notifies the sender about the buffer space available for each channel.
2. Data phase: the sender sends the data packets of all allocated channels starting with the highest priority channel.
3. Acknowledgement phase: the receiver acknowledges the data packets received.

The duration of each phase is the same for all nodes in the network although the number of data packets that can be sent can vary and may depend on the implementation, up to the maximum allowed by the duration of the data phase.

5.2.3.4 Fault Tolerant Mechanisms

The UTP provides the reliability already defined for BTP and SpaceWire-RT:

- Automatic retry of lost data
- Alternative or redundant network paths

In addition, it provides the following capabilities:

- Desynchronisation tolerant: Late arrival of acknowledgements is supported.
- Graceful degradation: Using multiple paths depending on the timeslot.
- Time-code errors: System goes silent if the local clock synchronization mismatches.

5.2.4 Experimental Apparatus

An experimental apparatus was developed to evaluate the UTP protocol operation and the performance of a software implementation using existing radiation-tolerant SpaceWire components. The selected space-qualified device was the Remote Terminal Controller (RTC) AT7913E [ATMEL 2010] which has an embedded LEON 2 processor [ATMEL 2010b].

The main characteristics of this software implementation written in C are:

- Performance: The prototype achieved 90Mbit/s of net user data rate using a timeslot duration of 135 µs, which allowed sending of up to six data packets of 256 bytes.
- Time synchronisation: It synchronises the schedule using Time-Codes. In case of time mismatch between local and Time-Codes timing it goes silent until it resynchronises on slot 0.
- Network congestion: Packets affected by unexpected sporadic congestion are allowed to arrive one slot late without immediately triggering a retry event, which in this case could increase the network congestion and worse the problem.

5.2.5 Protocol Evaluation

This section evaluates the improvements to the Quality of Service of UTP with respect to the scheduled mode of BTP. These are possible thanks to UTP being oriented only for scheduled networks.

5.2.5.1 Reliability

The UTP protocol has two important improvements in the reliability of packet transfer with respect to the scheduled mode of BTP:

1. The acknowledgements are received at the end of each slot so in case of error the data can be resent in the next valid slot. In BTP the acknowledgement was received in the next slot allocated to the same source-destination pair so the source could not resend the data until the following allocated slot.
2. UTP stores the scheduling in a node with a different path index for each slot. Each slot has associated a specific primary and redundant path. This allows a channel to use different primary and redundant paths depending on the slot.

The use of redundant paths to increase the reliability of the data transfer has not been evaluated before, so this section describes an experiment that focuses on this capability. It uses the experimental apparatus presented earlier and it was also demonstrated during the 14th SpaceWire Working Group meeting in February 2010 [Ferrer 2010].

The main objective of the experiment is to prove that UTP can deal with two basic kinds of errors:

1. Sporadic link errors are dealt with by the retry mechanism. They are nominally produced following the SpaceWire link error rate, which is very low, so this experiment forces them to happen using different techniques.
2. Permanent link failures are handled using redundant paths or links. These permanent failures are triggered in this experiment by removing the SpaceWire cables from the original setup while the experiment was running.

5.2.5.1.1 Experiment Setup

The experiment setup uses two sender nodes and two receiving nodes with two routers in between connected with three links. Sporadic link errors are injected using a STAR-Dundee Link Analyser [STAR-Dundee 2010b]. A STAR-Dundee Brick is used to configure and monitor the system and to inject unexpected congestion into the scheduled network.

Figure 5-21 and Figure 5-22 show the logical and physical configuration setup of the experiment, using the AT7910E SpW-10X routers [ATMEL 2010c].

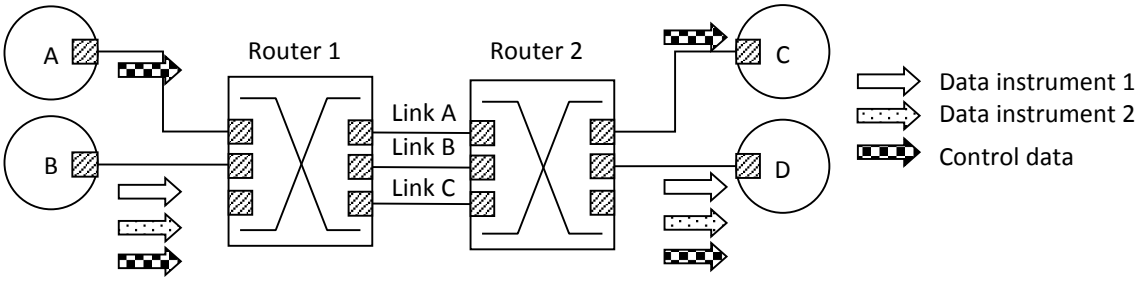


Figure 5-21: Experiment network setup

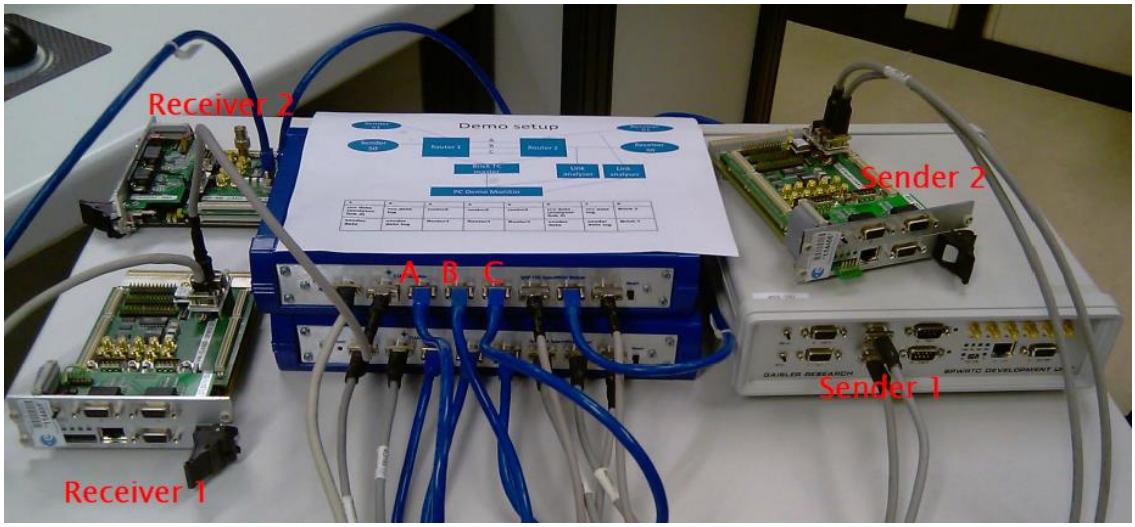


Figure 5-22: Experiment hardware setup

There are three types of data flows, two payload data and one command and control flow, which has higher priority and requires a higher level of reliability. The requirements of each channel are defined in terms of bandwidth, latency and reliability. Table 5-14 and Table 5-15 shows the list of channels of each sending node and its requirements.

Table 5-14: Channel requirements for sender node B

Channel #	Data type	Link Utilization	Latency	Reliability
#1	Control	1.65%	One timeslot	Support for two permanent link failures
#2	Payload data 1	49%	Two timeslots	Support for one permanent link failure
#3	Payload data 2	49%	Two timeslots	Support for one permanent link failure

Table 5-15: Channel requirements for sender node A

Channel #	Data type	Link Utilization	Latency	Reliability
#1	Control	1.65%	One timeslot	Support for two permanent link failures

The scheduling table of each node determines whether the set of channel requirements can be fulfilled. Table 5-16 and Table 5-17 shows the scheduling table for each node.

Table 5-16: Scheduling table for sender node B

Slot	High priority channel	Medium priority channel	Low priority channel	Path #	Primary path	Redundant path
0 + 4*N	#1	#2	#3	1	Link B	Link C
1 + 4*N	#1	#3	#2	2	Link A	Link C
2 + 4*N	#1	#2	#3	3	Link A	Link B
3 + 4*N	#1	#1	#2	2	Link A	Link C

Table 5-17: Scheduling table for sender node A

Slot	High priority channel	Medium priority channel	Low priority channel	Primary path	Redundant path
0 + 4*N	#1	-	-	Link A	-
1 + 4*N	#1	-	-	Link B	-
2 + 4*N	#1	-	-	Link C	-
3 + 4*N	#1	-	-	Link B	-

The scheduling mechanism of UTP is very flexible, allowing different priorities and paths for each slot:

- Even slots assign Data 1 flow a higher priority than Data 2 flow. Odd slots do the reverse. This gives them exactly the same priority and the same bandwidth assignment without having to implement a round-robin arbitration mechanism. Also, if one channel is stopped the other one can use its bandwidth.

- All channels use the three links available even if UTP only supports the definition of one alternative path per slot. If one or two link fails channels can continue sending data.

After the experiment setup is defined the results of error injection are presented.

5.2.5.1.2 Experiment Results

The first reliability test is the injection of sporadic link errors using the link analyser. Figure 5-23 shows a screenshot of the tool used to monitor the status of the experiment in the sending node A. After one link error has been injected a retry count is incremented in one of the channels but no user data errors are observed. Note that errors that affect the control channel are recovered in less than 100µs, which is the slot period. Therefore the message latency of the highest priority channel is deterministic, 100µs, and it is 200µs when an error occurs.

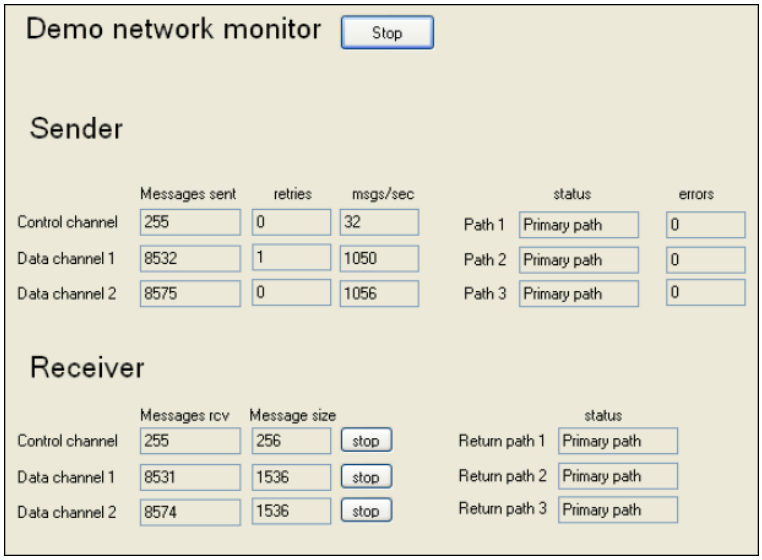


Figure 5-23: Protocol monitor screenshot when injecting a link error

Random packets are then injected into one of the routers using the STAR-Dundee Brick, simulating an unexpected network congestion. This produces the delay of the scheduled UTP packets, which results in multiple retry events showing in the experiment monitor.

However, UTP was designed to inject packets in a synchronous way but to receive in an asynchronous manner. Therefore it can be configured to tolerate temporal congestion due to this babbling idiot error injection. The data sent by UTP is not considered lost if the acknowledge is received before a timeout value, counted as a number of timeslots. The drawback is that it takes more time to detect and recover from a link error. During the experiment, the number of timeslots was increased until the retry events were no longer observed.

Finally, permanent link errors are evaluated. Permanent link failures trigger the automatic switching of the path. A permanent link error is detected when multiple retry events occur in a short period. The failed path is identified and disabled. In case an affected timeslot has no redundant paths the timeslot is disabled and the bandwidth provided lost. Affected channels using other timeslots and paths are still capable of sending data.

Figure 5-24 shows which redundant paths are activated when Link A is removed. Figure 5-25 shows the effect of an additional link B being also removed. One path entry is disabled and the bandwidth of the slots using this path entry is lost.

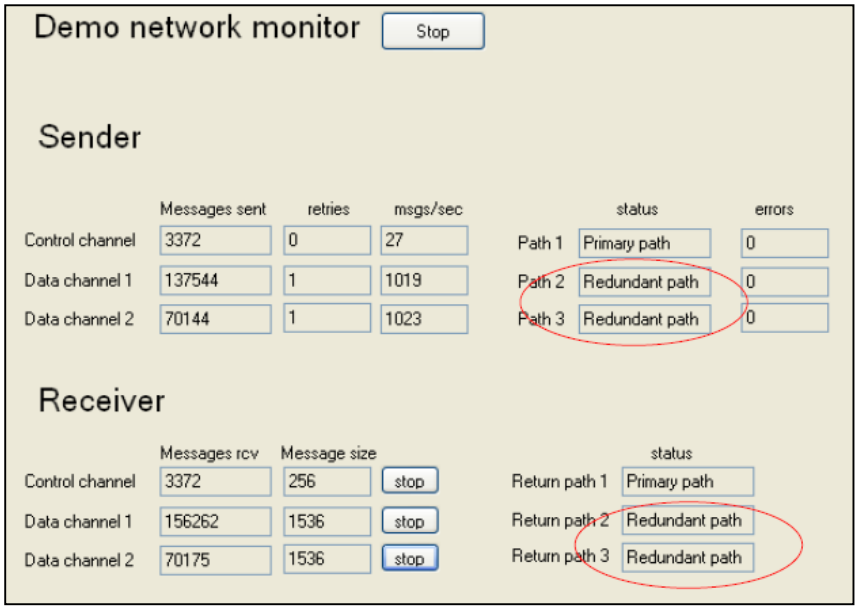


Figure 5-24: Protocol monitor screenshot when removing one link

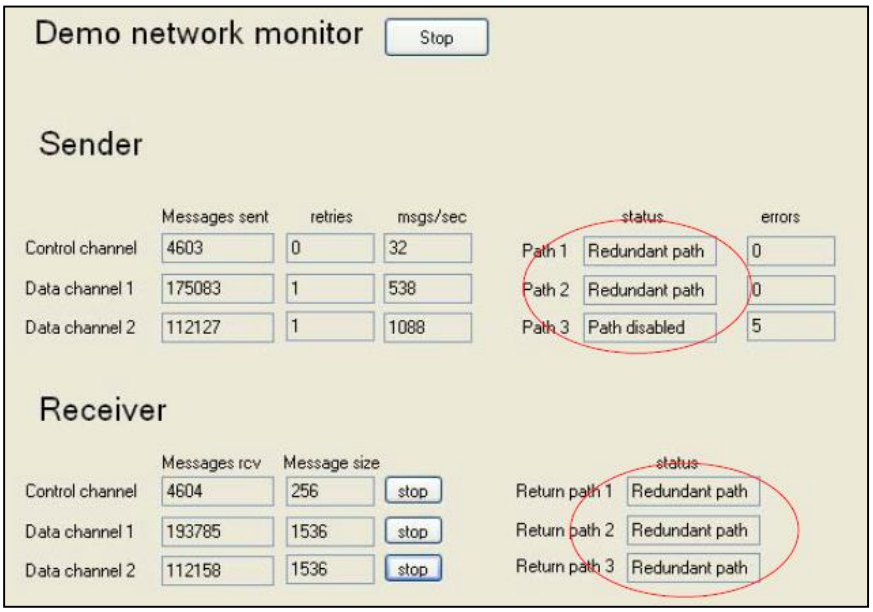


Figure 5-25: Protocol monitor screenshot when removing two links

5.2.5.2 Timely Delivery

The UTP protocol has been designed to improve the QoS performance of the scheduled version of BTP, especially for unidirectional data flows. Therefore, in this section this improvement is measured in a simple and easy to understand use case scenario.

5.2.5.2.1 Experiment Setup

Figure 5-26 shows a very basic spacecraft data-handling architecture where three nodes, one instrument, a Mass Memory unit and the onboard computer (OBC) are interconnected using a SpaceWire router.

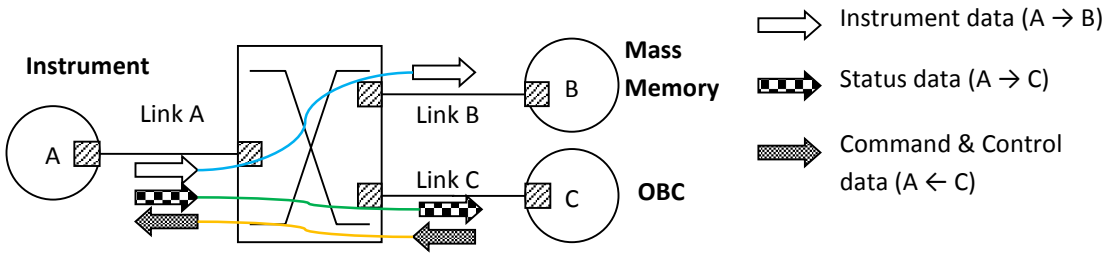


Figure 5-26: Simplified spacecraft data-handling network

There are three main data flows:

- Instrument data: High data rate flow going from the Instrument (node A) to the Mass Memory (node B) with a large message size representing, for example, frames of a video stream.
- Status data: The instruments (node A) sends periodic short messages to the OBC (node C) indicating the status of the instrument.
- Command and Control data: The OBC (node C) sends command messages to control the instrument (node A) when required.

Using BTP explained in the previous chapter, bidirectional slots equivalent to SpaceWire links would be allocated. Each link has a source-destination pair allocated for each timeslot. If the instrument data-rate requires between 1/3 and 2/3 of the link utilisation, a suitable network schedule table is shown in Table 5-18. Note that in this example the timeslot allocation repeats every 3 timeslots. In SpaceWire there are 64 timeslots available when using time-codes, so in this case the timeslot 0 is not used.

Table 5-18: Network Scheduling table for BTP

	Timeslot 3*N+**1**	Timeslot 3*N +**2**	Timeslot 3*N +**3**
Link A	**A → B**	**A ↔ C**	**A → B**
Link B	**A → B**		**A → B**
Link C		**A ↔ C**	

This schedule implies that command packets have a maximum latency of 3 timeslots, which is the case when the source generates the command just after the timeslot *3*n+2* begins. Note that the OBC cannot send packets to the Instrument in slots *3*N+1* and *3*N*+3, even if the instrument is not receiving data from the mass memory, only BTP acknowledgement packets.

UTP takes advantage of the fact that it allocates unidirectional data flows. Table 5-19 shows the scheduling table for UTP where each bidirectional link is divided in two unidirectional links, one for each direction.

Table 5-19: Network Scheduling table for UTP

	Slot 3*N+**1**	Slot 3*N +**2**	Slot 3*N +**3**
Link A→	**A → B**	**A → C**	**A → B**
Link A←	**A ← C**	**A ← C**	**A ← C**
Link B→	**A → B**		**A → B**
Link B←			
Link C→		**A → C**	
Link C←	**A ← C**	**A ← C**	**A ← C**

With UTP all slots can be allocated to the receiving direction of link A, so the maximum latency of command packets sent from the OBC is only one timeslot. This is one third of the value achieved using BTP.

5.2.5.2.2 Experiment Results

The maximum throughput that can be achieved also depends on the duration of a timeslot. Using the experimental apparatus described it is possible to use a minimum timeslot duration of only 100µs, but the throughput is increased if a timeslot of 200µs is used. Table 5-20 shows the experimental results for a maximum SDU size of 256 bytes. The efficiency of this software implementation is compared with the estimated value for a hardware implementation, which is assumed to achieve an inter-packet delay of a few microseconds. The efficiency of the hardware implementation is higher because it can send more data packets in the same timeslot period.

Table 5-20: UTP performance metrics

Timeslot / Minimum latency	# Data packets per slot	Maximum throughput	SW prototype efficiency	HW prototype estimated efficiency
100 µs	4	77Mbit/s	48%	81%
200 µs	10	103Mbit/s	64%	89%

5.2.6 Summary

This section has shown that the newly designed protocol called UTP improves, for scheduled networks, the performance of the QoS achieved by BTP, in terms of latency, throughput and reliability. The UTP was designed to improve the efficiency of BTP in a scheduled network with asymmetric unidirectional user data flows. The UTP implements an improved scheduling, segmentation and priority mechanisms, which in the network

scenario considered, provides better latency and throughput guarantees with a higher efficiency allocating timeslots.

UTP is especially suited to scenarios with asymmetric data flows, i.e. one side sends more data that it is receiving, which is common onboard spacecraft. UTP can allocate unidirectional data flows to unidirectional slots. A unidirectional slot only uses the bandwidth of one direction of each one of the links used. This improves the overall utilisation of the network with link utilisations up to 90% (see section 5.2.5.2.2).

Another advantage is that in the same timeslot, it can send multiple data packets belonging to different channels of the same source-destination pair. This usually improves the throughput. For example, in the scenario of Figure 5-21, if the data instrument 1 uses a portion of the bandwidth used by the data instrument 2 the timeslot can anyway be filled with the maximum data packets allowed. BTP, as it was defined, does not support that feature, unless some modifications are made to the protocol specification.

Finally, with UTP the sender receives the acknowledgement of the data packet sent in the same timeslot, so in case of an error, the retry is sent in the next allocated timeslot, which can be the next one. This can be important for critical command & control operations. Also, the flow control information is also received at the optimum time, reducing the receiver buffer size required to achieve the maximum throughput.

However there are two major drawbacks:

1. It is more complex to implement due to the subdivision of a timeslot into three phases. Most software implementations will use significant CPU resources and will be much less efficient than a specific hardware implementation.

2. It does not work well with some application protocols that require bidirectional data transfer even if the user data is a unidirectional data flow. The main case is when using RMAP read commands or RMAP write commands with acknowledgment. BTP can provide better performance than UTP in this case, but in chapter 6 a solution targeting this specific case maximises the performance when QoS is required.

5.3 Conclusions

This chapter has presented two different transport layer protocols, BTP and UTP, which provides Quality of Service for current generation of SpaceWire devices. Reliability is provided using acknowledgments and retry capabilities. Timely delivery is provided using segmentation and TDM techniques.

BTP uses bidirectional channels to cope with any type of user data flow and can work in asynchronous or synchronous mode.

UTP is specialised to be more efficient with one typical use case for onboard spacecraft networks, a synchronous network that mixes control and payload data and uses data flows that are unidirectional.

Chapter 6: Cross-layer Quality of Service for SpaceWire

This chapter will explore the possibility of using the cross-layer optimisation paradigm to increase the efficiency and reduce the cost of implementing QoS protocols. Cross-layer design refers to protocol design done by actively exploiting the dependence between protocol layers to obtain performance gains. Figure 6-1 shows two of the most-used cross-layer techniques, the cross-layer integration, which combines the capabilities of multiple layers into a single protocol, and the cross-layer feedback, which enables status information of one layer to be used in another layer.

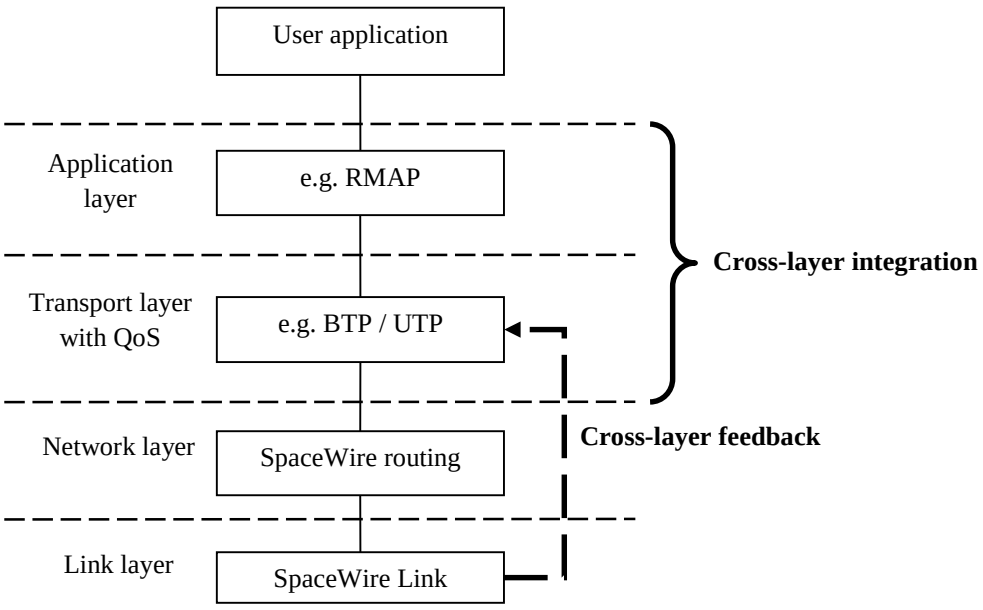


Figure 6-1: Cross-Layer QoS techniques with RMAP and SpaceWire link layer

Section 5.2.6 has pointed out that UTP seems quite inefficient when carrying RMAP packets. The reason is that RMAP is a bidirectional protocol that has both capabilities of a transport and application layer. It implements the acknowledgements of a transport protocol and the remote memory addressing of an application protocol. Therefore, the SpaceWire Working Group considered the possibility of using only RMAP protocol to provide the required QoS to SpaceWire networks. The biggest advantage is that there are

already multiple devices that support RMAP, most of them in hardware implementations [ATMEL 2010][ATMEL 2010c]. Using an application protocol like RMAP to provide transport layer capabilities with QoS is a clear example of cross-layer integration. This will be discussed in the first part of this chapter, with the development of a cross-layer scheduler based on RMAP. It will be evaluated with an experimental apparatus using VHDL hardware language and implemented in a FPGA.

The second part of this chapter deals with the use of the cross-layer feedback technique. The author of this thesis has explored the possibility of using the link-layer flow control mechanism of SpaceWire to provide status feedback to the transport layer dynamically. This kind of cross-layer optimisation is widely used with wireless networks to optimise the Quality of Service achieved [Liu 2004]. However, to the knowledge of the author, this idea has never been used in wormhole switching networks like SpaceWire. Here it will be analysed and evaluated the possibility to use the status of the link-layer flow control mechanism for error detection and network arbitration of competing data flows.

6.1 RMAP Scheduler Protocol

As previously explained, the Remote Memory Access Protocol (RMAP) is a transaction based protocol with one node, the Initiator, sending an RMAP command to read or write data to registers in a memory address located in another node, the Target. The main idea of this section is to implement QoS by scheduling the network, as is done with UTP and BTP, but using RMAP packets instead of a specific designed protocol and packet format.

RMAP provides error detection using acknowledgments, it performs the most usual operations (read or write) with user messages of any size, and it is usually already implemented in typical SpaceWire systems.

The RMAP scheduler can be used for two different use cases:

1. To provide timely delivery and latency guarantees for RMAP transactions used by a user application.
2. To provide QoS guarantees for the sending of arbitrary user messages. This is done using RMAP commands even if the user did not consider the use of the RMAP protocol. The user messages are carried in the data field of the RMAP packet (SDU).

In order to evaluate this cross-layer integration idea, a prototype called RMAP scheduler was developed, details of which are given in 6.1.5. The objective was to provide similar or better Quality of Service metrics with less cost when using RMAP protocol and current generation of SpaceWire devices. This time the prototype was implemented in hardware using VHDL. This provided performance values for hardware schedulers and allowed to better trade off parameters such as the timeslot period. The result of this work was used

as an important input for the standardization efforts of SpW-D [Parkes 2010], a protocol based on using a RMAP scheduler like the one developed here.

6.1.1 Concept

The basic idea of an RMAP scheduler is that RMAP packets are sent at specific moments following a global synchronization that ensures that two different transactions do not use the same network resources at the same time. The simplest implementation is to use a local scheduler at each node that transmits one RMAP command just after the reception of a Time-Code. Figure 6-2 shows a simple example with two nodes that transmit read and write RMAP commands to a third node using a shared link.

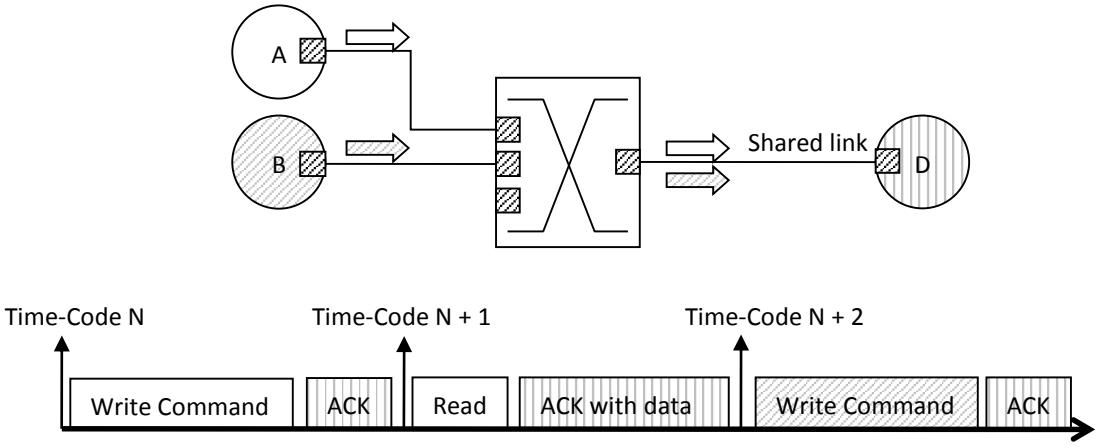


Figure 6-2: Scheduling read and Write RMAP commands

The RMAP acknowledgements provides reliability and the scheduling provides timely delivery, but there is an important Quality of Service element that it is not provided, the end-to-end flow control mechanism.

However, when the destination address is not a FIFO but an addressed-based memory element, it is safe to assume that the destination will be ready to handle the data of a write operation. In other words, it does not require end-to-end flow control mechanism to avoid stalling or rejecting a receiving packet. The case of a receiving FIFO becoming full can

be handled by issuing an RMAP reply with a specific error code. With the RMAP scheduler the sender assumes that the destination always has space and it will produce an error if that it is not the case, which should only happen under non-nominal conditions.

6.1.2 Requirements

The requirements for the RMAP scheduler protocol follows the guidelines of section 5.1.1.1 and the list of requirements of section 5.1.1.2 related with synchronous networks, with the exception that the retry mechanism is optional. The key requirements are:

- Scheduling of RMAP commands with different priorities.
- Detection and recovery of Time-Code synchronization errors.
- Detection and recovery of temporary network congestion.
- Segmentation of user messages and conversion of a single RMAP command containing a large data unit into multiple smaller RMAP commands.
- Remote configuration and protocol operation using RMAP.

The last point also includes the capability to automatically notify a remote node when an error occurs. This is especially useful when dealing with passive nodes without embedded CPUs, which cannot make a complex decision when an error occurs.

6.1.3 Design Considerations

The main objective of the RMAP scheduler is to allow a simple and efficient scheduling of RMAP messages with high reliability. An RMAP message consists of one or more RMAP packets. Therefore, this section will discuss the following issues:

- Adapt the concept of channels used for BTP and UTP to the RMAP protocol.
- Trade off the duration of timeslots for the RMAP scheduling

- Support remote control and error notification.

When dealing with the previous points, the following considerations related with Time-Division Multiplexing techniques will be taken into account.

- The duration of a timeslot has to be traded off to achieve a high data rate for payload data and a low latency for command and control operations.
- High priority event-based messages are difficult to schedule.
- Errors in the network can produce timing violations in the global schedule and induce unexpected contention.

6.1.3.1 RMAP Scheduler Channels

Channels in UTP and BTP were based on sending and receiving FIFOs where the user applications wrote and read data. RMAP scheduler aims to integrate the RMAP protocol with the transport protocol. It appears to be simple to directly relate a channel to the characteristics of the RMAP commands that send a message. Furthermore:

- When dealing with a large user message that cannot fit into a single timeslot, a segmentation mechanism is needed to split it into multiple smaller RMAP read/write commands that are sent across several timeslots (that may not be consecutive).
- It is possible that several channels can be active concurrently when messages are large and need to be split into multiple timeslots.
- A channel needs to be retriggered each time an RMAP message is sent, and it needs to be reconfigured when the message parameters change.

The channel configuration can be performed by the node or by external network manager using RMAP. The Quality of Service of each channel depends on how timeslots are

allocated. It is also possible to further control the message rate or throughput by using different slot allocations, depending on the current epoch, which can increase the efficiency of the system.

6.1.3.2 Network Scheduling

Network scheduling is more efficient when the data traffic is known and periodic. It is difficult to schedule event-based messages that require low latency, especially if they use little bandwidth and are rarely generated. With a simple local schedule where each message must be allocated to a different slot, this rare, high-priority message has to be allocated to at least one slot. This slot will be unused most of the time.

The solution is to implement a priority mechanism on top of the schedule table like that performed with BTP. Note that the more flexible priority scheme of UTP is not implemented because it does not work well with the remote error notification scheme proposed in section 6.1.3.4. Therefore the proposed solution is to just set a different priority for each channel of a node.

To send critical sporadic messages efficiently a high priority channel is configured to use the same timeslots that have been already allocated to a long payload message using a lower priority channel. The long payload message is sent using multiple segments, one for each timeslot. When the control message must be sent it will be sent in the following allocated timeslot even if the long payload message is still active. The number of timeslots required must take into account the total bandwidth required. For example, there could be one channel for a payload message that requires six timeslots and two channels for two control messages that need half timeslot each, requiring a total of seven timeslots per epoch.

It has been stated that in scheduled networks multiple transactions can take place at the same time providing that they do not use the same network resources, i.e. they do not produce contention. For BTP the network resource was a SpaceWire link and for UTP a unidirectional link, i.e. only one direction of a SpaceWire link is used for the slot allocation.

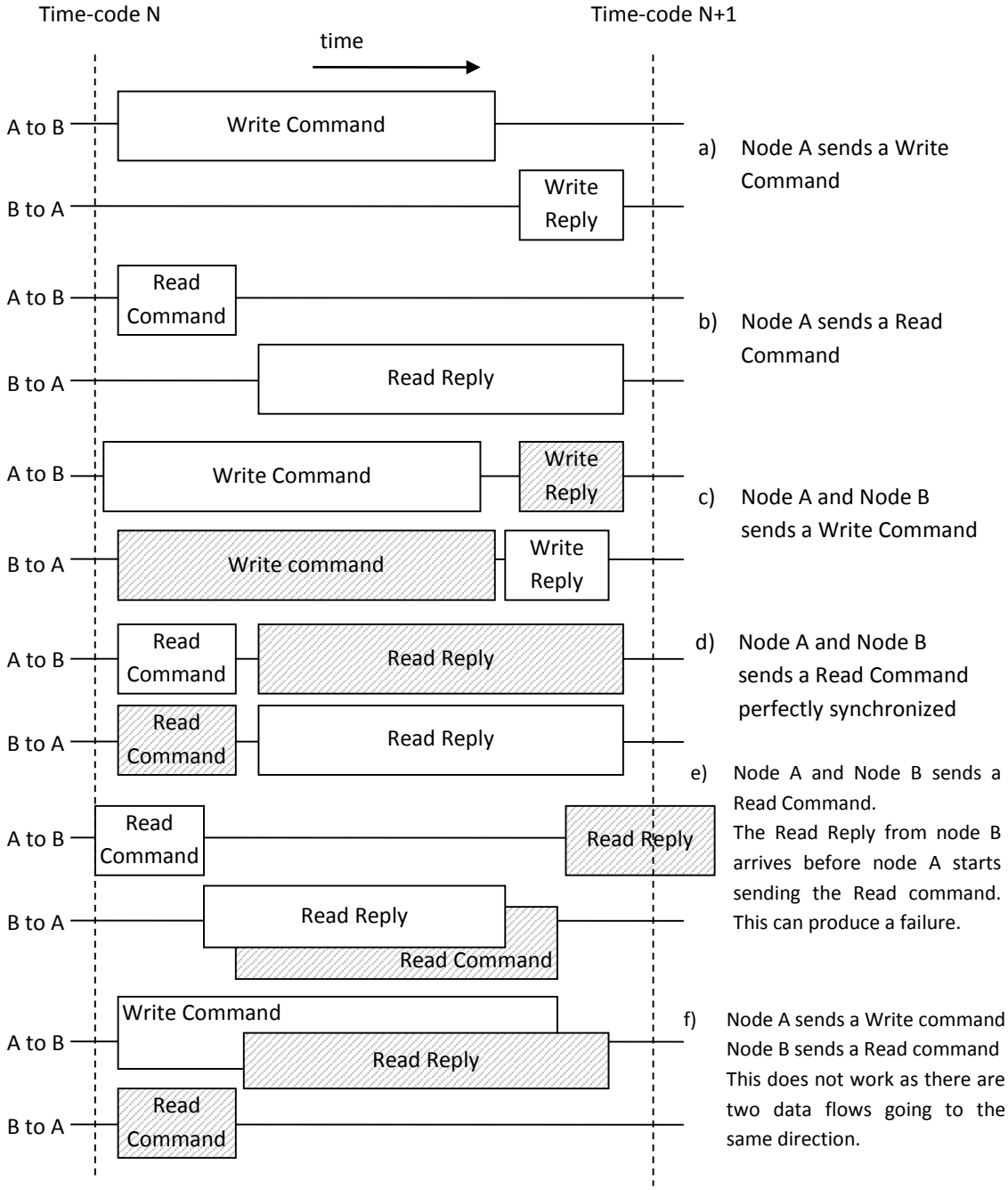


Figure 6-3: RMAP commands going in different directions of the same link

For the RMAP scheduler it is possible to schedule the transaction of two RMAP commands going in different directions if they both use the same RMAP command type. However, the implementation should ensure that the RMAP commands are sent before any RMAP reply is sent. Figure 6-3 shows each possibility.

6.1.3.3 Timeslot Duration

The duration of a timeslot determines the maximum size of the SDU contained in a RMAP write command or RMAP read reply. The size of the SDU is specified in the data length field of the RMAP packet. As the RMAP scheduler implements a segmentation mechanism, the SDU size determines the size of each segment. For the timeslot duration trade off the following assumptions have been made:

- Only one RMAP transaction is allowed to be executed in a single timeslot.
- The maximum link speed is set 200 Mbps, which gives the best performance for space-qualified hardware.
- RMAP commands are processed by dedicated hardware in the target node. It is then possible to have a fixed protocol overhead of around 15-20 µs per timeslot. This includes the protocol header, the network latency and the processing time.
- The maximum data length for an RMAP packet is set to a multiple of a power of two and it should be larger than a typical control message for space applications. Typical values would be 256, 512 and 768 bytes.

The estimated timings related to protocol overheads of a timeslot is shown in Table 6-1 considering 200Mbps and a maximum distance of 4 hops [ATMEL 2010c].

Table 6-1: Estimated timings overheads of the RMAP scheduler

Timeslot phase	Estimate duration
Time-Code reception(T_{TC})	1.2 µs
Network packet latency (T_N)	2.5 µs
RMAP headers (T_H)	1.5 µs
RMAP command processing (T_{PC})	5 µs
RMAP reply processing (T_{PR})	5 µs

The minimum duration of a timeslot is determined by the simple equation (6.1), where the S is the link speed (Mbps) and L_{SDU} is the data length of the SDU (bytes).

$$T_{TS} = T_{TC}+T_H + T_N * 2 + T_{PC} + T_{PR} + \frac{L_{SDU} * 10}{S} \qquad (6.1)$$

In section 5.2.2.3 the timeslot duration for the UTP protocol was optimised for the maximum protocol efficiency when sending one segment. Here is considered instead the overall message size, which may consist of multiple segments.

For each timeslot duration the maximum SDU size is computed that can fit in the timeslot with the restriction of being a multiple of 128 bytes. Then, the maximum and the minimum protocol efficiency of a message consisting of multiple segments is computed. The maximum efficiency is achieved when the message size is exactly a multiple of the SDU size and the minimum efficiency occurs when the last segment of a message has a data length of only one byte, i.e. the message size is a multiple of the SDU size plus one. Figure 6-4 shows the minimum and maximum efficiency for each timeslot period assuming a minimum message size of 2 Kbytes. A logarithmic tendency line has been added for the minimum and maximum values.

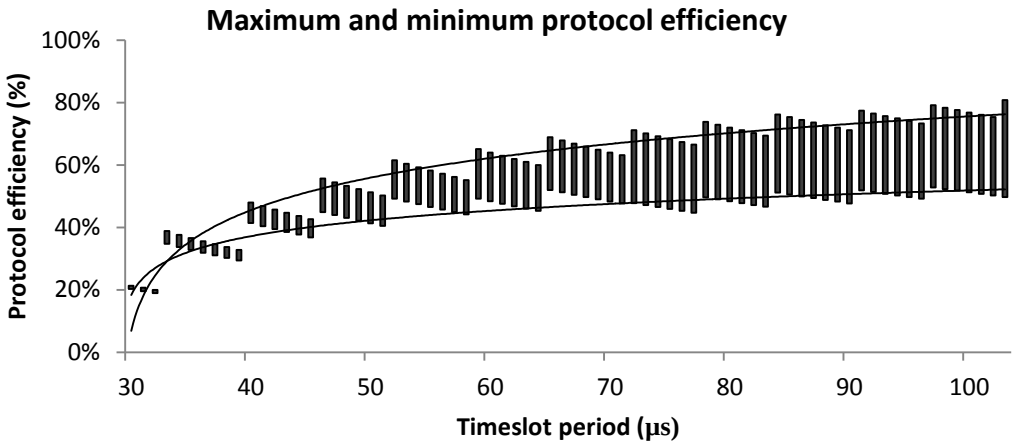


Figure 6-4: RMAP scheduler efficiency versus timeslot period

The SDU size for the timeslots periods should also comply with the CCSDS CUC format, which requires the timeslot period to be a power of two division of one second [CCSDS 2002].

Table 6-2: Protocol efficiency depending on timeslot and SDU size.

Timeslot period (µs)	SDU size	Minimum efficiency	Maximum efficiency
40	384 bytes	41%	48%
50	512 bytes	41%	51%
64	768 bytes	45%	60%
80	1152 bytes	48%	72%
100	1536 bytes	51%	77%

Results in Table 6-2 show that the minimum efficiency does not increase significantly with longer timeslots, so it is not useful to use a timeslot period higher than 100µs as this increases the latency of a small control messages. Note that any control message requiring low latency should fit into a single SDU.

A good selection criteria for the timeslot duration is to choose the smallest timeslot period (but higher than 40µs) that has an SDU size larger than the largest control message used.

This ensures the minimum latency and the maximum accuracy for bandwidth allocation using timeslots.

If a high-protocol efficiency is required for high data-rate data flow, it is possible to use a technique that is called here multi-slotting. The idea was presented in the SpaceWire Working Group and it basically allows one large data packet PDU to be sent across multiple consecutive timeslots. The acknowledgement PDU is received at the last timeslot. With this technique it is possible for certain channels to use larger SDUs and reduce the impact of the protocol overheads timings. Another advantage is that it allows the use of a network with different link speeds. For example, a node connected to a 50Mbit/s link would use four consecutive slots that will be equivalent to a single slot of a node working at 200Mbit/s.

6.1.3.4 Error Handling

There are two important errors related with scheduling networks that were not previously discussed for the UPT and BTP protocols:

1. A Time-Code is lost or arrives late or too early.
2. A SpaceWire packet is not sent before the end of the current timeslot because of unexpected network congestion.

The first error can be handled by using a local clock with each node. If there is a discrepancy between the timeslot timing provided by the network and the local clock, the system should go to a fail-safe mode and discard the current timeslot.

The second error can be produced when using the current generation of SpaceWire routers not designed for scheduling networks. Specifically, the SpW-10X induces congestion

when a packet is sent to a link that has failed. This can be solved using the technique explained in section 4.2.10.1.

Still, it is possible to have network congestion due to some faulty unit. The protocol should ensure that RMAP commands do not fail because of temporary network congestion. This can be easily done with the following rules:

1. If a SpaceWire packet is not sent before the end of a timeslot it is allowed to be sent in the next timeslot instead of other RMAP commands scheduled in this timeslot.
2. No packets will be sent in the next timeslot after the congested packet has finished being sent. This idle timeslot ensures that there is enough time for the routers to spill other blocked packets from the network and avoid error propagation due to cascade network congestion. This procedure is similar to the one used by TCP to handle network congestion.

Therefore, a channel related with an RMAP command should not be considered in error if an RMAP reply is not received within the same timeslot. Instead, it is considered that the command was not received or the reply was lost if the RMAP reply is not received after two more timeslots have elapsed. This gives time for the faulty packets to be removed by the SpaceWire routers and the valid RMAP commands to arrive and be processed. Only the faulty packet will not produce an acknowledgement so its associated channel will be the only one disabled, preventing error propagation.

When a channel is disabled because of an error it is useful to notify the event to a remote node. One simple idea is to activate certain channels only when another channel presents an error. The new activated channel can contain a message with the error condition. This

can also be used for redundancy mechanisms. A retrial mechanism can also be implemented by the user application by retriggering the channel in error.

6.1.4 Protocol Specification

This section summarises the operation of the RMAP scheduler protocol based on the design considerations explained before.

6.1.4.1 PDUs Format and Channel Sequence

The PDU format of packets sent by the RMAP scheduler are the ones defined by the RMAP protocol. However, the RMAP scheduler requires the utilisation of the most significant byte (MSB) of the "Transaction ID" field (see Figure 6-5) so the user application can only make use of the less significant byte of this field for its own purposes. The MSB of the "Transaction ID" is used to handle, with multiple channels, the segmentation capability of the protocol.

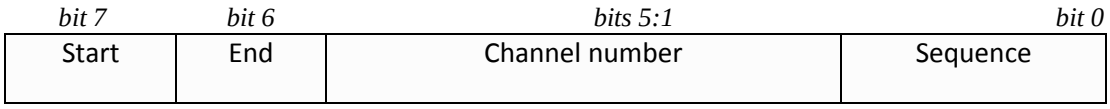


Figure 6-5: MSByte of the RMAP Transaction ID used by RMAP scheduler

Table 6-3: RMAP Transaction ID fields used by RMAP scheduler

Field	Description
Start	Set when the PDU is the first segment of the user RMAP message.
End	Set when the PDU is the last segment of the user RMAP message.
Channel number	Specifies the channel number related with the sequence number.
Sequence	1-bit sequence number of the channel specified

Figure 6-5 shows the different fields explained in Table 6-3. The Sequence number only needs to be one bit as there are no outstanding transactions, i.e. a send and wait for

acknowledgement scheme is used. The receiver of an RMAP write command (RMAP target) should store the last sequence number for each channel. This is used by the RMAP target to discard duplicated RMAP write commands that are resent when an error occurs. Note that this mechanism is not required if the RMAP target stores received RMAP SDUs in a memory location that can be overwritten, as in this case the reception of a duplicated SDU does not lead to data corruption.

6.1.4.2 Channel Operation

A channel describes a single RMAP message configuration (i.e. the header including the destination) and its allocated timeslot numbers. It implements a transparent segmentation layer and provides sending status and error reporting. Multiple channels can be active at the same time. A channel is enabled when it has been configured with all RMAP parameters required and it has not send all its data. A long message may use multiple slots, each one containing one segment of data. Each channel has a channel number and a priority level. For each timeslot the highest priority channel that is enabled is used. Once the highest priority channel has finished sending its message, a lower priority channel is used. For simplicity, an implementation can directly relate the channel number with the priority level, with the channel 0 having the highest priority.

A channel is only disabled if a reply packet is not received after a programmable number of slots. It is then assumed that the RMAP command was not received or the reply was lost. Therefore, if there is congestion while transmitting a packet, i.e. the packet has not been sent at the end of a timeslot, or if the RMAP reply did not arrive in the same slot, an error is reported but the channel is not immediately disabled. Optionally, a channel can be activated automatically when another one has been disabled. This can be used for remote error notification or for redundancy mechanisms.

6.1.4.3 Time Synchronization

As done with BTP and UTP, time synchronization is achieved by distributing timecodes across the network using SpaceWire Time-Codes, which have lower network latency than SpaceWire packets. The period between the arrival of consecutive Time-Code codes is measured with an internal clock and compared with the expected value. In case of discrepancy, the system does not trigger the sending of any packet and it only reports the error to the user application (early or late Time-Code arrival). The system also automatically resynchronises without further user interaction when the period between consecutive Time-Codes matches the expected timeslot period. Figure 6-6 shows this mechanism.

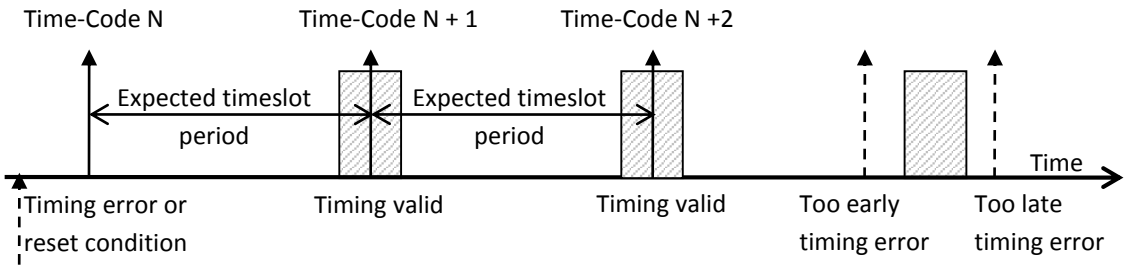


Figure 6-6: Timing synchronization mechanism

6.1.5 Experimental Apparatus

The capabilities of the RMAP scheduler protocol were evaluated with a hardware implementation in a Xilinx Virtex II and IV FPGAs [XILINX 2011] [XILINX 2011b]. This allowed to obtain precise timings for the real operation of this scheduled protocol.

The ESA RMAP IP Core, which was designed by the University of Dundee, was used to implement the RMAP protocol. This allowed to minimise the development time and focus only on the specific features of the RMAP scheduler.

6.1.5.1 Architecture

Figure 6-7 shows the architecture of the experimental apparatus inside the FPGA and how it can be connected to a remote application node via a SpaceWire router. For simplicity, the host application was not developed and the remote application was used instead to configure and run the experimental apparatus using RMAP packets.

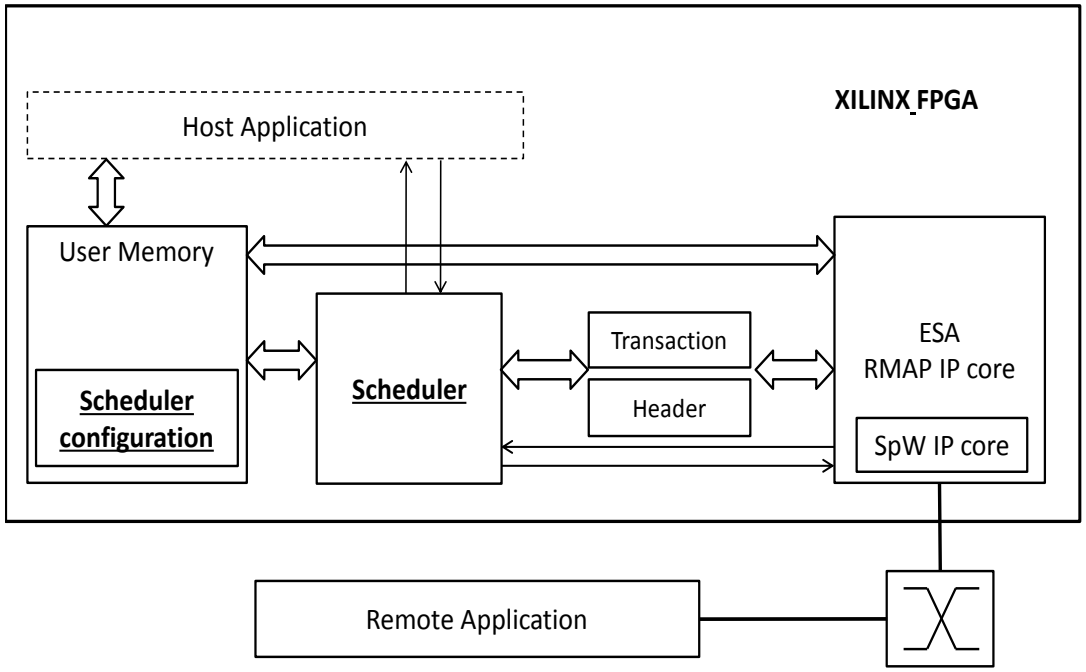


Figure 6-7: RMAP Scheduler implementation architecture.

The RMAP IP core has an RMAP target and an RMAP initiator modules that serves two different uses:

1. The RMAP target is used to configure and control the RMAP scheduler from the remote application node without requiring a Host Application. It is also used to process RMAP commands sent by another node implementing the RMAP scheduler.
2. The RMAP initiator is used to send the RMAP commands corresponding to segments of the different RMAP scheduler channels.

The RMAP scheduler is configured by writing to the Scheduler Configuration memory space, which contains the configuration for multiple channels and the global timing setup. The Scheduler module generates the RMAP information required by the RMAP initiator module to send the next segment of a user message corresponding to a channel. This basically consists of setting the RMAP command header and transaction data pointers. The segment to be sent will depend on the status of the highest priority active channel allocated to the next slot. When a Time-Code is received, the Scheduler module also triggers the RMAP initiator to send the next scheduled RMAP command packet.

Figure 6-8 shows a simplified block diagram. When a Time-Code is received, its timing is check by the Time-Code Handler. If it is a valid Time-Code, the Error Detection module checks if the next scheduled channel and the system is not in error. The Transaction Trigger module then triggers the ESA RMAP IP Core while the Channel Status Updater updates the state of the channel, including the segmentation status. Finally, the Transaction Generator selects the channel that will be active in the next slot, based on the schedule table, the priority level and the status of each channel. This module also generates the RMAP transaction associated to the selected channel.

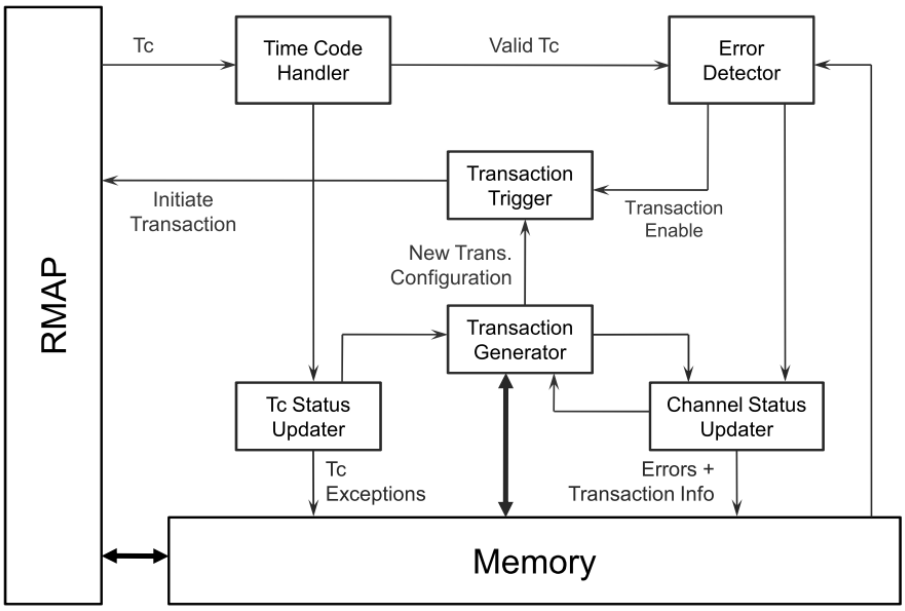


Figure 6-8: RMAP Scheduler block diagram

The module can be configured at run-time with any Time-Code period and each channel can have a different message size and segmentation size value.

6.1.5.2 Performance and Cost

The key performance metrics of a scheduler is the time it takes to send the scheduled packet since a Time-Code was received. The part of the delay due to the Scheduler module developed is only 300ns which is the time it takes to validate the Time-Code received and trigger the RMAP initiator to send the RMAP command previously configured. Then the RMAP IP core then needs around 3 µsec to start sending the RMAP command.

Regarding the implementation cost, it can be measured by the area of the FPGA used.

Table 6-4 shows the values obtained for the whole prototype which are compared against the ESA RMAP IP Core. The results show how the part corresponding to the network scheduler module roughly requires less than a quarter of the original RMAP IP core resources.

Table 6-4: Resource used by RMAP and the Network Scheduler

FPGA Virtex 2 Xilinx Resource	RMAP IP Core	RMAP IP Core plus Network Scheduler	Additional Cost
Slice Flip-Flops	3002	3868	+ 29 %
4-input LUTS	8722	10498	+ 20 %
Occupied Slices	5023	6207	+ 24 %

6.1.6 Protocol Evaluation

This section evaluates the Quality of Service provided by the RMAP scheduler implemented in the experimental apparatus described before. The evaluation is done from two different perspectives:

1. Analysis of the reliability and timely delivery provided by the RMAP scheduler in a generic use case where the RMAP protocol was not initially intended to be used by the user application. For this analysis a simple data-handling network is considered.
2. Analysis of the improvement of the QoS achieved by the RMAP scheduler when the user already considered use of the RMAP protocol for its application. The network setup is the same as the one used for BTP and UTP in order to compare the different QoS performance.

6.1.6.1 Reliability and Timely Delivery

This section evaluates the reliability and timely delivery of the RMAP scheduler protocol for a simple data-handling network where the RMAP protocol is not used by the user application.

6.1.6.1.1 Experiment Setup

The experiment setup is shown in Figure 6-9. It consists of three FPGAs, implementing the experimental apparatus, which simulates an instrument, a payload processor and an event data logger unit. A PC interacts through a STAR-Dundee Brick with the different elements but only for control and monitoring purposes. There is also a mass memory unit emulated with an RMAP target enabled device, where the instrument and a payload

processor can either read or write data. All units are connected through a SpaceWire router. The data event logger is used to store high priority event messages [Ferrer 2010b].

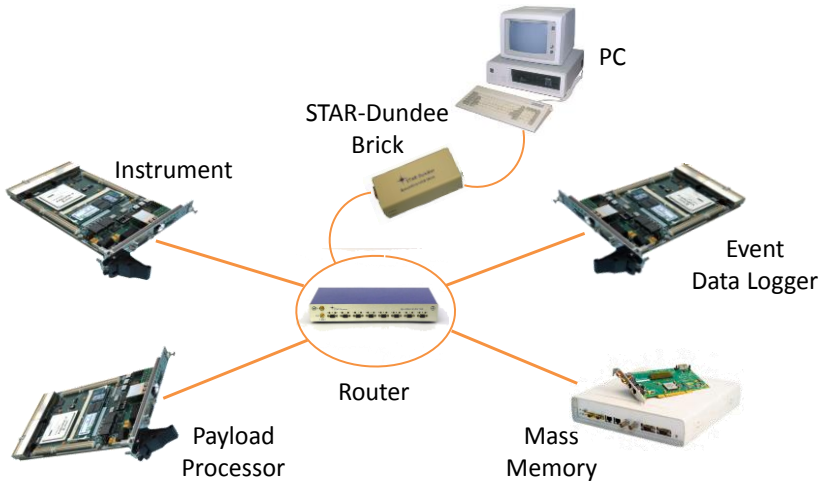


Figure 6-9: Setup of the RMAP scheduler experiment.

The schedule setup for this experiment is simple and it is based on using odd and even timeslot numbers for different source-destination pairs. The instrument sends during even timeslots RMAP write commands to the memory. The payload processor alternatively reads and writes to the Mass Memory during odd slots simulating a data compression application.

The Instrument unit has an input hardware trigger that, when asserted, activates a channel of the RMAP scheduler and triggers the sending of a high priority message to the event logger. The RMAP write command containing this message can be sent at any timeslot as no other data flow is using the same path. The Instrument unit also has an additional channel that is activated only when the high priority message fails to be delivered. Figure 6-10 shows all the data flows described.

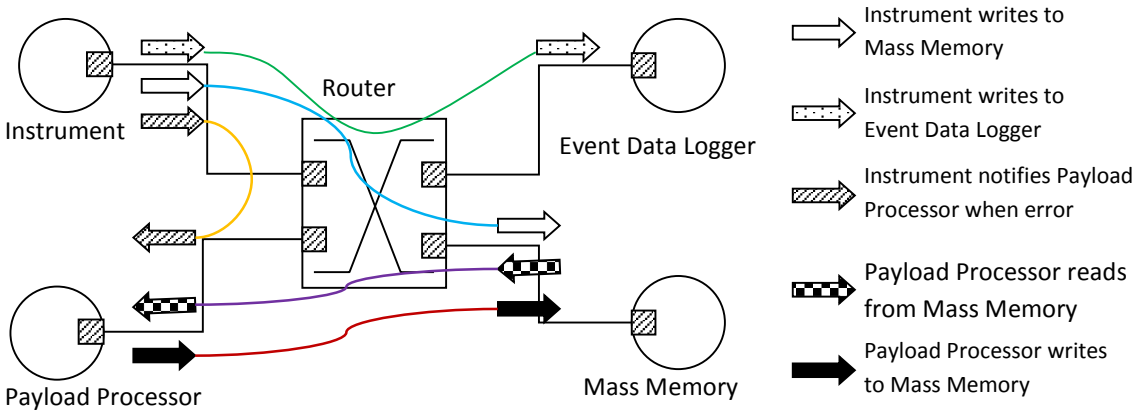


Figure 6-10: Data flows of the RMAP Scheduler experiment.

The timeslot period is set to 50 µs. The link speed is set to only 100Mbit/s due to hardware limitations of the FPGA, and the SDU size is set to 256 bytes. For testing purposes, the SDU size is smaller than the maximum possible as there is margin for a larger value without requiring a longer timeslot period.

6.1.6.1.2 Experiment Results

The experiment executed successfully, following the expected deterministic behaviour. The following tests were performed:

- The data received in the Mass Memory matched the data to be sent by the Instrument.
- The time between the reception of a Time-Code and the sending of the first byte of the RMAP command was measured with a value of only 3 µs.
- When a pulse was injected to the Instrument hardware trigger an RMAP write command was sent to the Event Data Logger unit.
- If the link to the Event Data Logger was disconnected before a pulse was injected to the Instrument hardware trigger, an error report message was sent automatically to the Payload Processor.

These results successfully demonstrate the deterministic behaviour of the RMAP scheduler and the new error reporting features provided. A basic QoS metric, the latency of the high priority message, was also measured with detail. Figure 6-11 shows a screenshot of an oscilloscope that illustrates the measurement of the latency and jitter of the high priority channel. The left side shows the pulse introduced to the input trigger of the Instrument that causes the activation of the high priority channel. The oscilloscope is configured to be triggered with this pulse. The right side shows a sequence of lines, which are multiple traces of clock pulses created by the Event Data Logger each time a high priority message is received.

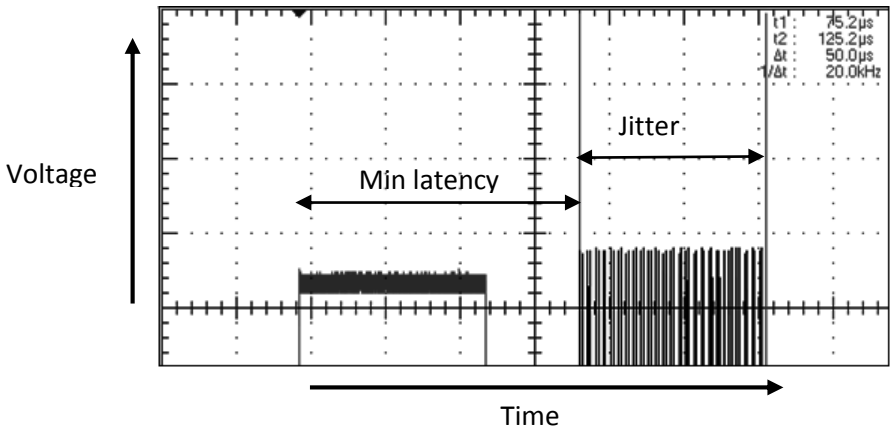


Figure 6-11. Measurement of latency and jitter for the high priority channel.

Note that the period of time determined by the two cursors at the right is equal to the duration of a timeslot. This means that the jitter of the high priority message is one timeslot. The minimum latency is one timeslot plus the time required to receive the RMAP write command since the start of the current timeslot. The maximum latency is the minimum latency plus the jitter which is one timeslot. The jitter is expected to be one timeslot because the message can be triggered to be sent at any moment. The additional latency of one timeslot is due to the fact that in this implementation the channel selection is not computed when the Time-Code arrives but in the previous timeslot.

6.1.6.2 Optimisation for RMAP Applications

The RMAP scheduler protocol has been designed to improve the QoS metrics of UTP when using the RMAP protocol for data transfer. Therefore, this section will use the same scenario as section 5.2.5.2.1 but using write RMAP commands, see Figure 6-12.

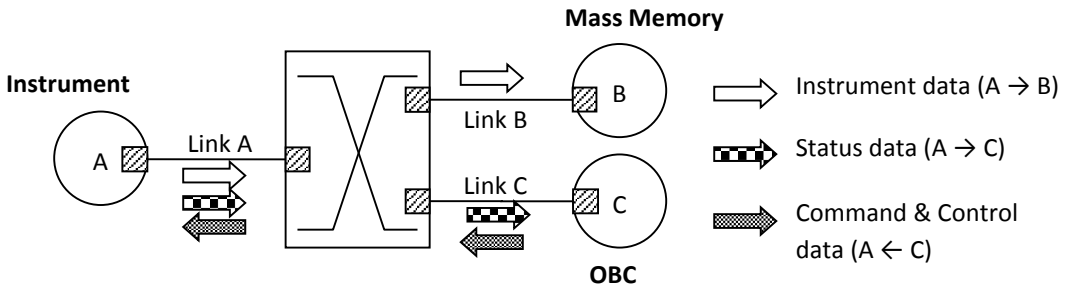


Figure 6-12: Simplified spacecraft data-handling network

As stated in section 5.2.6, with UTP if the user would want to send RMAP write commands with the acknowledgement option, it would require to schedule the acknowledgement packets of this application protocol. Note that RMAP acknowledgement mechanism cannot be compared with the acknowledgements generated by a transport protocol like UTP or BTP. The RMAP acknowledgements also contain application specific information, such as error codes related to RMAP command authorization including invalid memory addresses.

BTP can accommodate these RMAP acknowledgements using the same schedule table shown in Table 5-18. However, for UTP the schedule of Table 5-19 does not work when RMAP write acknowledgements are considered and instead a scheduling table similar to the one for BTP is required. Table 6-5 shows this working scheduling for UTP, where the path A → C comprises two channels, one for sending status data and the other one for sending the acknowledge packets of the received command & control data. This applies

for the reverse path A ← C. With UTP these two channels can send data packets in the same slot. On the other hand, with BTP only one channel can sent data packets in one timeslot, so BTP needs to decide between these two channels every 3 slots.

Table 6-5: Network Scheduling table for UTP using RMAP

	Slot 3*N+**1**	Slot 3*N +**2**	Slot 3*N +**3**
Link A→	**A → B**	**A → C**	**A → B**
Link A←	**A ← B**	**A ← C**	**A ← B**
Link B→	**A → B**		**A → B**
Link B←	**A ← B**		**A ← B**
Link C→		**A → C**	
Link C←		**A ← C**	

Now with the RMAP scheduler there are two important advantages:

1. There is no need to allocate a slot for the RMAP acknowledgement packets
2. The instrument node can decide at the start of each slot if it sends an RMAP write command with Instrument data to the Mass Memory node or if it sends an RMAP write command with status data to the OBC node. This reduces the maximum latency of the higher priority status data to a single slot, which is very useful when notifying that an error in the instrument has occurred.

With the previous considerations, the scheduling table for the RMAP scheduler is shown in Table 6-6.

Table 6-6: Network Scheduling table for RMAP scheduler protocol

	Timeslot 3*N	Timeslot 3*N +**1**	Timeslot 3*N +**2**
Link A→	**A → B** **A → C**	**A → B** **A → C**	**A → B** **A → C**
Link A←	**A ← C**	**A ← C**	**A ← C**
Link B→	**A → B**	**A → B**	**A → B**
Link B←			
Link C→	**A → C**	**A → C**	**A → C**
Link C←	**A ← C**	**A ← C**	**A ← C**

Table 6-7 shows a comparison between the BTP, UTP and the RMAP scheduler of the maximum latency for each data flow, using the scheduling tables presented. The RMAP scheduler provides the best results. Note that the latency here is measured from the instant the data is generated asynchronously to the time it can be sent in an allowed timeslot. Therefore it indicates the additional delay due to scheduling and does not include the transmission time.

Table 6-7: Maximum latency of BTP, UTP and RMAP scheduler

	BTP	**UTP**	**UTP with RMAP**	**RMAP scheduler**
Scheduling table	Table 5-18	Table 5-19	Table 6-5	Table 6-7
Instrument data	2 timeslots	2 timeslots	2 timeslots	1 timeslot
Status data	3 timeslots	3 timeslots	3 timeslots	1 timeslot
Command and control	3 timeslots	1 timeslot	3 timeslots	1 timeslot

Note that it is assumed that node A does not generate instrument data if there is pending status data to be sent.

Now it is considered the worst case end-to-end delay between the time the user data is generated at the source asynchronously and the time the acknowledgement generated by the destination is received by the source.

Table 6-8 shows this deterministic worst case end-to-end delay when the user application does not use the RMAP protocol. In this case the RMAP scheduler encapsulates the user protocol within RMAP packets. The acknowledgements are sent within the same timeslot with UTP and the RMAP scheduler, and in the next available timeslot with BTP.

Table 6-8: Maximum end-to-end delay without using RMAP

	BTP	**UTP**	**RMAP scheduler**
Scheduling table	Table 5-18	Table 5-19	Table 6-7
Instrument data	3 timeslots	2 timeslots	1 timeslot
Status data	6 timeslots	3 timeslots	1 timeslot
Command and control	6 timeslots	1 timeslot	1 timeslot

Finally, it is considered the worst case end-to-end delay when the user application does use the RMAP protocol. Table 6-9 shows that the RMAP scheduler is much more efficient than other protocols because the RMAP acknowledgement is sent in the same timeslot as the RMAP command is sent. BTP is bidirectional so it sends the RMAP acknowledgement in the next timeslot. UTP needs to wait until the timeslot allocated to the reverse unidirectional channel.

Table 6-9: Maximum end-to-end delay using RMAP

	BTP with RMAP	**UTP with RMAP**	**RMAP scheduler**
Scheduling table	Table 5-18	Table 6-5	Table 6-7
Instrument data	3 timeslots	3 timeslots	1 timeslot
Status data	6 timeslots	6 timeslots	1 timeslot
Command and control	6 timeslots	6 timeslots	1 timeslot

6.1.7 Summary

The RMAP scheduler was designed to improve the efficiency of UTP when the RMAP application protocol is being used by the user. However it is also useful in cases where the RMAP protocol was not intended to be used because it allows to dynamically choose the destination node at the beginning of a timeslot.

The protocol uses cross-layer concepts to benefit from the error detection capabilities of RMAP. The design also incorporates other advanced error detection and notification mechanisms. When combined, this solution delivers high performance scheduling and segmentation capabilities to RMAP applications with a small additional cost.

An example of the improved end-to-end delay of RMAP commands sent by the user, in comparison with the other protocols developed (BTP and UTP) is shown in Table 6-9.

In order to determine suitable values for the timing margins of this synchronous protocol, a hardware prototype of the RMAP Network Scheduler was developed in VHDL using the ESA RMAP IP core. The RMAP Network Scheduler was successfully validated through simulations and implemented in Virtex II and IV FPGAs. The FPGA version proved its capabilities in a realistic scenario with several RMAP devices such as the Remote Terminal Controller [Habinc 2007] interacting in real time.

The RMAP scheduler designed was used for the design and the evaluation of the SpaceWire-D protocol [Parkes 2010]. The main difference with the RMAP scheduler implemented is the additional capability of the SpaceWire-D protocol to support multiple RMAP transactions in one timeslot. This allows use of longer timeslots when higher link utilisation is required by some software-based scheduling schemes.

6.2 Quality of Service using Link-Layer Feedback

This section describes two novel cross-layer optimisation techniques between the transport layer with QoS and the link layer of SpaceWire networks. The transport layer can acquire out-of-band status information by using specific characteristics of wormhole switching networks like SpaceWire. It is a cross-layer feedback technique where the transport layer uses specific resources of the link layer for its own operation. The study is based on SpaceWire but it could be applied to other wormhole switching networks with link-layer flow control.

Figure 6-13 shows the two new transport protocols developed in this section, the Link Backpressure Protocol (LBP) and the Network Arbitration Protocol (NAP), and how they acquire link-layer status information from the SpaceWire link layer.

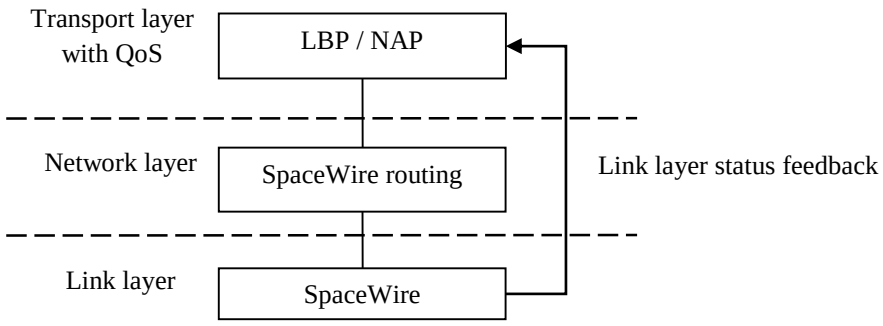


Figure 6-13: QoS using Cross-layer Feedback from the SpaceWire link layer

6.2.1 Link Backpressure Protocol

The Link Backpressure Protocol (LBP) aims at providing a reliable transport protocol with end-to-end flow control without requiring the receiver node to send acknowledgments or flow control status packets.

6.2.1.1 Concept

The key idea is to use the link-layer SpaceWire flow control characters, called FCTs, which travel in the opposite direction to the SpaceWire data characters, to provide end-to-end status information from the receiver to the sender.

These link-layer flow control characters are only valid in point-to-point SpaceWire links. When there is space in the receiving buffer of a SpaceWire codec for eight more data characters, a flow control character (FCT) is sent to the other end of the link. If there is no space, no FCTs are sent and when the FCT credit is zero, no more data characters can be sent, creating a backpressure effect. Figure 6-14 shows the case when the destination only has buffer space for a single FCT.

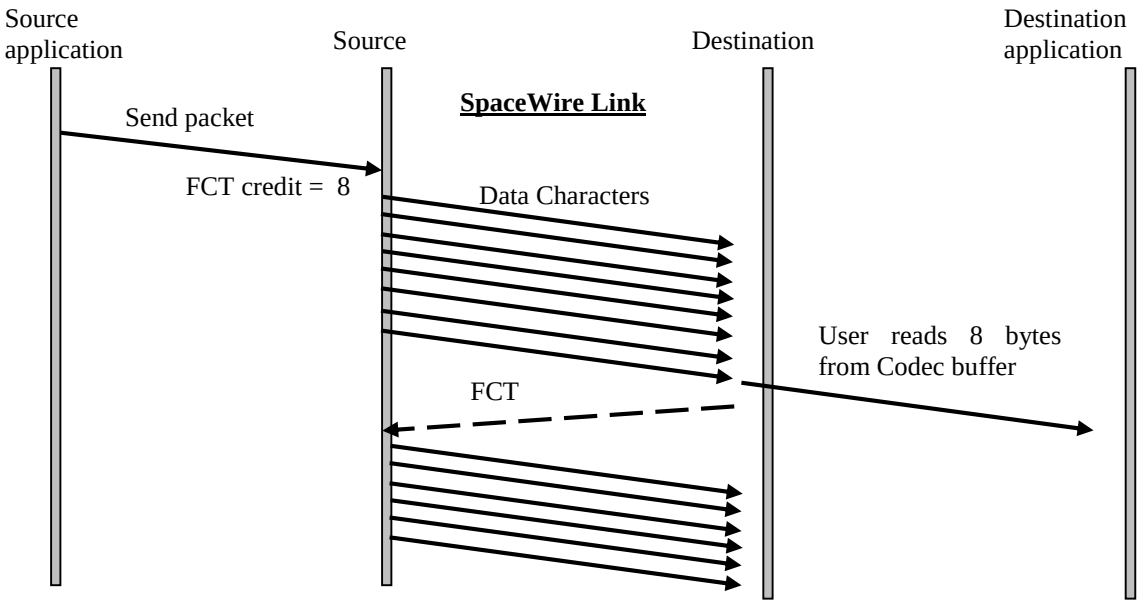


Figure 6-14: SpaceWire Flow Control Tokens (FCTs)

A transport layer protocol applies to an end-to-end connection that can consist of multiple point-to-point links and some SpaceWire routers in between. Each hop in the network has a SpaceWire Codec with its own buffers, but they are interconnected when a packet flows from one end to the other. In Wormhole Switching networks, routers do not store the complete packet but only as many data characters as defined by a single flow control unit (one FCT for SpaceWire networks).

Therefore, although the FCT characters are not directly transmitted between one link and another, the backpressure effect can still go from the packet destination node to the source if the packet is larger than the storage capacity of the intermediate buffers within the routers. In other words, the data source can notice that the destination has stopped sinking data, (backpressure effect) before the source has finished transmitting the packet.

The key difference with other networks based on packet switching is that in SpaceWire networks the buffers in the routers are very small and there is a small number of them, so for medium-sized packets the transmitting side has not finished sending the packet when the receiving side starts to receive the packet. As stated, if the packet is larger than the total size of the router buffers through the network path, the receiver can stop the reception of the packet and the sender can detect this event. The sender will then stop the transmission of the packet because the link buffer in the closest router will become full. The sender will not be able to continue the transmission of the packet until the receiver starts to issue new FCTs.

During nominal operation the destination is expected to constantly issue new FCTs when it is receiving a packet. The action of stopping the reception of a packet provides a single bit of information to the source. The destination can stop after a specific number of bytes

are received so the instant in which the destination stops can provide more bits of information to the source.

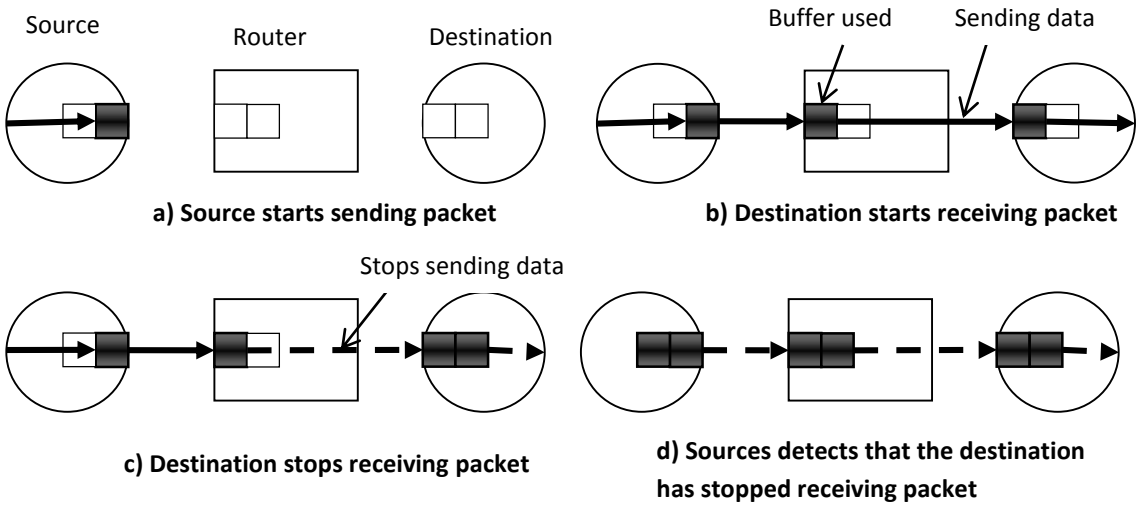


Figure 6-15: Link-layer flow control pauses the packet transmission

Figure 6-15 shows the sequence of events that occur when the destination stops sinking data. The small boxes indicates the buffer space within the router or node. When the packet is being transmitted and received only a portion of the buffering space is used. When the receiver stops receiving this packet, the buffers become full starting from the destination node. When the router buffers become full the sender node detects that the packet has stalled and has to stop transmitting data characters of the packet. This event is the key information used for the Link Backpressure Protocol or LBP.

The main benefit of the mechanism described is that it can provide an acknowledgement notification that it is immediate and cannot be lost as if it was a packet. Other protocols like BTP require a timeout mechanism in the sender for the case an acknowledgment packet is lost. The other advantage is that in scheduled networks it is not required to reserve timeslots or bandwidth for the acknowledgement packet.

6.2.1.2 Design Considerations

The design of a protocol that makes use of the previous concept needs to take into account the possible errors that may occur in the network. Below is a list of possible errors and their implications when the link-layer flow control is used for information transfer between the destination and the source.

- Network congestion: The source detects that the packet has stalled and cannot transmit data for some time until the network congestion is solved. As the packet has not yet arrived at the destination and the network buffering is limited the amount of bytes transmitted will be small in this case.
- Link disconnection: This can cause some temporary network congestion by some SpaceWire routers. After the congestion the packet will be spilt and in any case the router will continue the spilling until the EOP marker is received.
- Invalid destination or node failure: The packet can be immediately spilt or become stuck at the receiving node until the router spills the blocked packet.

The protocol should be designed so that none of the previous error causes can create the same behaviour as the destination node when it wants to acknowledge the reception of a packet. The following rules should be applied by LBP because it cannot be produced by any of the previous error conditions:

- End-to-end notification procedure: Given the value B as the number of bytes that can be stored in the network buffers along the longest network path, the destination can notify an event to the source by stopping the reception of a packet after at least B bytes have been received and before there are less than B bytes left to be received before the EOP of the packet. The stopping duration should be

longer than the time required to fill the network buffers which is shorter than the transmission time of B bytes.

The rule described can only be applied if the packet length is larger than $2 * B$ bytes, i.e. the double the sum of the network buffer's capacity along the path from the source to the destination. There are also other restrictions:

- The link speed should be the same for all links of the network
- The protocol must ensure that the destination hardware does not stop receiving data arbitrarily.

The end-to-end notification procedure described is used in LBP to notify the reception of one or more packets, i.e. acknowledgment information, or to provide end-to-end flow control information. The following techniques can be used to obtain this protocol status:

1. Use the number of bytes transmitted before stopping as bits of information, taking into account the accuracy of the mechanism.
2. Use the number of times the sender has had to stop sending data as bits of information.
3. Measure the amount of time that the source cannot send data as bits of information.

Note that the more often the destination stops receiving and the more often it stays stopped, the less efficient is the protocol. The first technique can also be problematic as it makes the system less robust. The timing of the stop event is better used to check for the proper protocol operation instead of providing information regarding the status of a communication channel.

Another issue of the proposed notification technique is that the CRC covering the SDU of a packet should be checked before the receiver provides the acknowledgment to the sender. There are two possible solutions:

1. The receiver acknowledges the received packet in the following packet that it receives.
2. The packet contains B or more filling zeros after the CRC of the SDU and before the EOP.

The second solution is preferred even if it is less efficient. The first option is much more complex and difficult to implement and has the problem that it requires the sender to send a packet even if it has no data to send in order to acquire the acknowledgement of the previous packet.

Finally, it should be noted that it is possible to increase the efficiency of the protocol by using a sending window where the sender only requests an acknowledgement to the receiver every certain amount of packets. For simplicity this optimisation will not be considered for the evaluation of the LBP protocol.

6.2.1.3 Protocol Specification

The PDU format of the LBP protocol is new and it is shown in Figure 6-16. The Data Length field gives the number of bytes of the SDU, the Flags field is described in Table 6-10, the Fill field contains only zeros and the other fields have the same meaning than in BTP and are described in Table 5-5.

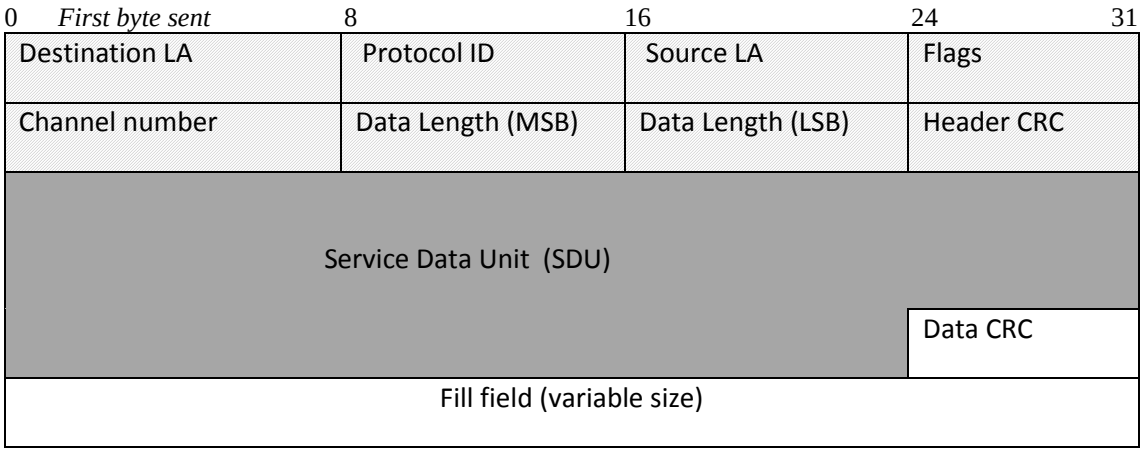


Figure 6-16: BTP Protocol Data Unit (PDU) format

Table 6-10: LBP Flags field

Bits	Name	Description
0	Reset flag	Indicates that the receiver should reset the packet sequence number and stop receiving data during 1ms after the header is received. The receiver should also count the length of the Fill field.
1	Start flag	Indicates start of message
2	End flag	Indicates end of message
3	Check flag	If it is set, the receiver should pause the reception when the first byte of the Fill field is received, if the data CRC is correct or if there is no buffer space. If it is not set, the filling zeros field is not included in the PDU. This bit is always set in the protocol version evaluated.
4-7	Sequence number	It is incremented each time a new packet is sent, independently of the channel used.

The LBP protocol can implement the retry mechanism with a Send-and-Wait or a Go-Back-N scheme depending on the value of the Check flag. If the Check flag is always set, the receiver will always pause the packet reception if the data CRC is correct. For the Go-Back-N it will only be set by the source node when the sending window is full.

When the source needs to send a message to a new destination the following procedure should be followed, so the protocol parameters are initialised.

1. The source starts sending a PDU with a zero length SDU and with the Reset flag set. The source should continue sending zeros after the header CRC (Fill field) until the destination stops receiving (during 1ms) when it receives the LBP header. The source should measure the number of bytes transmitted (B) when it detects the link-layer backpressure.
2. The source node then sends the EOP when an FCT is received after at least half a millisecond of not receiving one. When the receiver receives the EOP it counts the length of the Fill field and use it to compute the time it has to wait when it stops receiving a packet, based on the link speed set.

   In case the source node does not support the sending of a dynamic sized packet, it can do the same job with two PDUs with the reset flag set. First has a 1kbyte Fill field length and the second one has the Fill field length of B bytes.

This procedure ensures that both the destination and the source nodes obtain the protocol parameter B required for nominal operation. During nominal operation, the destination and the source should follow the rules described in Table 6-11 and Table 6-12.

Table 6-11: LBP destination node rules

Check order	Event type	Short receiving pause between header and data CRC	Short receiving pause between data CRC and EOP
1	CRC error in the header	No	No
2	Sequence error	Yes	Yes
3	Invalid channel	Yes	Yes
4	No receive buffer space	Yes	No
5	Valid header invalid data CRC	no	no
-	Valid header and data CRC	no	Yes

Table 6-12: LBP source node rules

Short pause detected between header and data CRC	Short pause detected between data CRC and EOP	Action
No	No	Resend packet (link or CRC error)
Yes	No	Channel receive buffer full: 1) Discard packet 2) Send a packet from another channel
Yes	Yes	Resend previous packet (invalid validation) If it fails again, reset connection (fatal error)
No	Yes	Packet validated. Send next packet.

6.2.1.4 Experimental Apparatus

The LBP protocol described was implemented in a software prototype using the radiation tolerant LEON processor of the Remote Terminal Controller (RTC). The objective was to evaluate if this protocol could be implemented in software.

The SpaceWire interface of the RTC does not allow monitoring by software of the number of bytes sent until the complete packet is sent. This was a fundamental issue for the implementation of LBP and it was solved by using the LEON trace buffer. A software driver for LBP was developed that was able to look at the LEON memory space read by the SpaceWire interfaces when a packet stored was being transmitted. This allowed to monitoring of the bytes sent at any time by polling the driver status periodically. Another issue is that the software implementation of the receiver needs to hold the packet header to check its content. This hold period is smaller than the one specified in Table 6-12 to avoid confusion.

The most important aspect of the RTC implementation was the discovery that the internal SpaceWire buffering of RTC was larger than expected which required the increase of the minimum parameter B.

Note that due to time restrictions only the Send-and-Wait scheme was implemented.

6.2.1.5 Protocol Evaluation

This section evaluates the benefits provided by the LBP protocol and the QoS aspects of reliability and timely delivery of messages.

6.2.1.5.1 Reliability

The protocol operation and its reliability were validated using the simple topology shown in Figure 6-17.

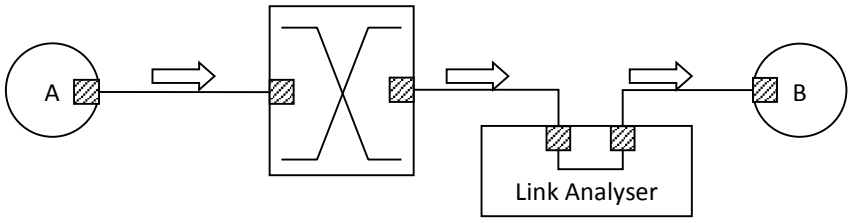


Figure 6-17: Network setup for evaluating LBP protocol

The first step was to measure the total amount of buffering between the sender node A and the receiver node B by forcing the receiver to stop sinking data when the header was received and then measure in the sender node A the number of bytes transmitted.

The value obtained was 274 bytes, which was larger than expected due to the internal buffering of the SpaceWire interface of the RTC. This value can be estimated using equation (6.2), where H is the header size of LBP, Bt, Br are the buffer size in transmission and reception, and N_{FCTs} is the number of FCTs sent during SpaceWire link

initialisation. The equation (6.2) allows computation of the network buffering size B for any number of links or network hops once the buffer sizes of the RTC are known.

$$B = H + N_{links} * (N_{FCTs} * 8 + Bt_{router}) + Bt_{RTC} + Br_{RTC} \\ = 8 + 2 * (4 * 8 + 5) + Bt_{RTC} + Br_{RTC} = 274\ bytes \quad (6.2)$$

As stated, this value is the number of additional bytes with no valid data that need to be sent between the data CRC and the EOP marker, the *fill* field in Figure 6-16. They are used to fill the network buffers. The time required to fill the network buffers is always less than the transmission time of this amount of bytes. Therefore with the link speed set at 200Mbit/s, the packet halt duration T_{halt} was set in receiving node B to around 15µs which gives some margin for the jitter of the RTC software.

Once node B was configured, node A was triggered to send LBP packets continuously. The data received was checked for the data pattern used for the test. The link analyser was used to inject link errors and check the protocol operation. The data was received correctly and the protocol resent the data lost when injecting sporadic link errors. However, when injecting link errors continuously the system failed after some time. The exact reason for this failure has not been determined. Given the significant complexity of the prototype, the problem was not investigated further as it had little impact on the evaluation of LBP.

Table 6-13 shows the user data rate achieved with the experimental apparatus based on the RTC for different SDU sizes. Note that the packet size (PDU) of the LBP is the data size of the SDU plus the header plus the network buffering size plus some bytes of margin.

Table 6-13: LBP throughput achieved with RTC prototype

SDU data size (bytes)	LBP PDU size (bytes)	Throughput
256	544	33Mbit/s
512	798	60Mbit/s
1024	1310	91Mbit/s

The throughput can be estimated by computing the latency using equation (6.3), where L_{SDU} is the SDU data size, T_{pkt} is the inter-packet delay, T_h is the delay required to process the header, and T_{halt} is the packet halt duration. Equation (6.4) gives the actual value of the throughput, which is the SDU size in bits divided by the latency.

$$D = \frac{(H + L_{SDU} + B) * 10}{S} + T_{pkt} + T_h + T_{halt}$$
$$D_{(L_{SDU}=256)} = \frac{(8 + 256 + 274) * 10}{200} + 20 + 10 + 15 = 72\ \mu s \quad (6.3)$$

$$TH_{(L_{SDU}=256)} = \frac{L_{SDU} * 10}{D} = \frac{256 * 10}{72} = 35Mbit/s \quad (6.4)$$

It is clear that the throughput achieved is quite low using the RTC. However, with a hardware implementation there is no internal buffering in the nodes and there is no need to stop sinking data when the receiver examines the LBP header. Therefore, T_h is zero, T_{pkt} is smaller and Bt_{RTC} , Br_{RTC} are zero, reducing T_{halt}. With these assumptions the LBP latency, throughput and protocol efficiency (link utilisation) with software and hardware implementations for different SDU sizes can be estimated. Figure 6-18 shows the protocol efficiency derived for the RTC and for a typical hardware implementation.

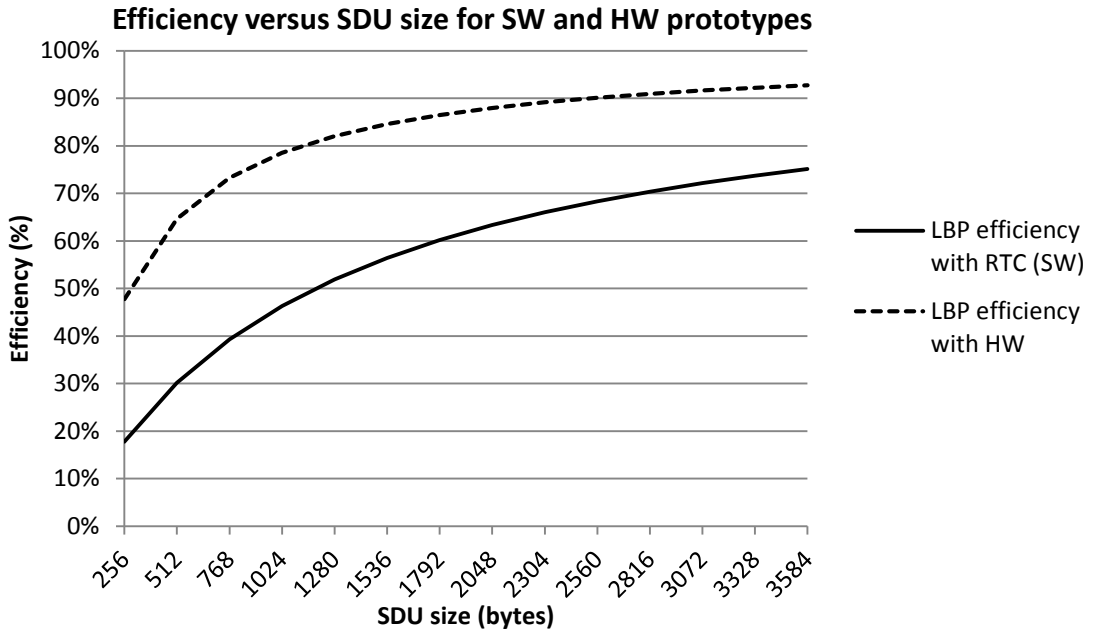


Figure 6-18: LBP protocol efficiency or link utilisation versus SDU size

The latency is the time it takes to send a LBP packet including the transmission time so it defines the minimum timeslot when LBP is used in scheduled mode.

Figure 6-19 shows the efficiency of the protocol for hardware and software implementations depending on the timeslot period. The timeslot period is unique in all networks so it is interesting to know the efficiency of the protocol for both hardware and software implementations of LBP. The results show that the timeslot needs to be at least 150µs to achieve a minimum of 60% protocol efficiency or link utilisation when the link speed is 200Mbit/s.

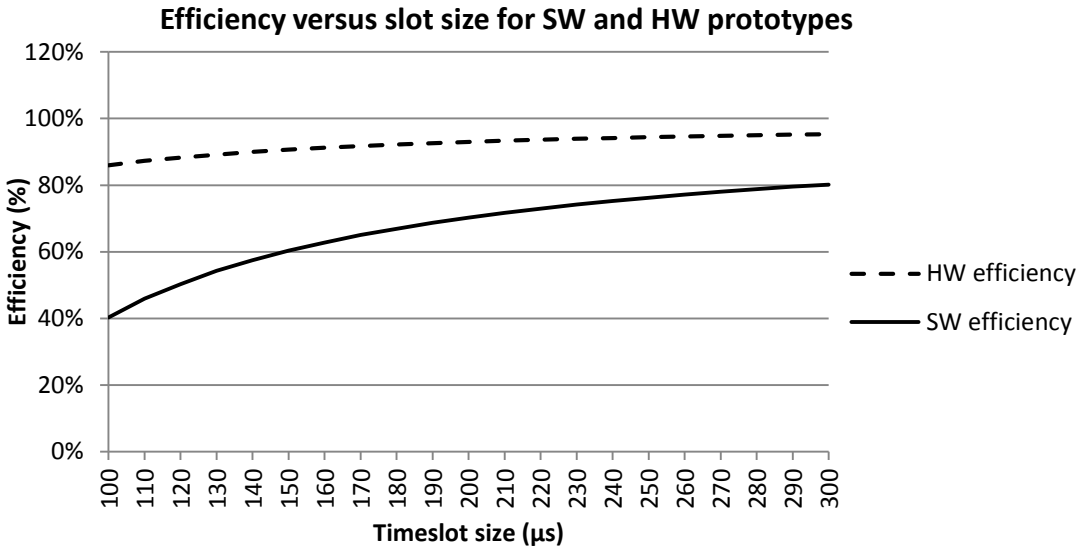


Figure 6-19: LBP protocol efficiency or link utilisation versus timeslot size

6.2.1.5.2 Timely delivery

The reliability and the throughput of LBP has been already evaluated. Now this section evaluates the benefits that the scheduled mode of LBP provides in terms of timely delivery when the same scenario used to compare the RMAP scheduler, the UTP and the BTP protocols is considered. For convenience this scenario is showed again in the following Figure 6-20.

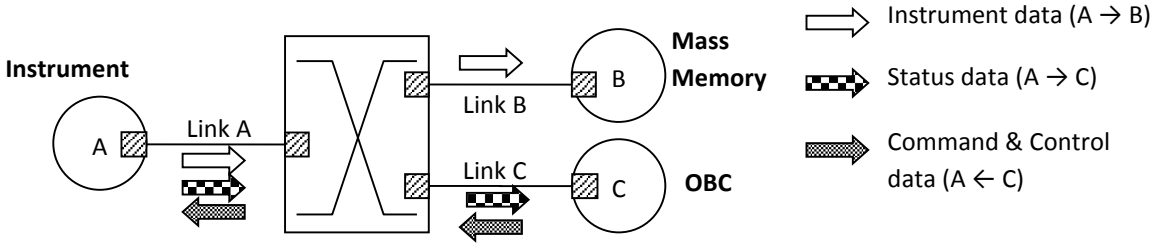


Figure 6-20: Simplified spacecraft data-handling network

The major advantage of LBP is that it does not require the reception of acknowledgements or flow control packets. This has two main benefits:

- LBP allocates unidirectional slots like UTP and RMAP and does not allocate bidirectional slots or links like BTP. This gives more efficiency and flexibility to the design of the network scheduling table.
- LBP can allocate, to the same timeslot and from the same source, channels sending data to multiple destinations. In other words, the source can decide at the beginning of a slot, where to send data, between multiple target nodes provided that the appropriate unidirectional links were allocated to this slot. This also applies to the RMAP scheduler.

With these premises, the scheduling table of LBP is the same as that for the RMAP scheduler (see Table 6-6), with two main differences:

- It does not use the RMAP application protocol.
- It provides end-to-end flow control.

6.2.2 Network Arbitration Protocol

The Network Arbitration Protocol (NAP) aims at providing low latency guarantees for high priority messages using a TDM mechanism that does not require configuration of schedule tables. Therefore it is especially efficient for networks with unknown, non-periodic traffic where an asynchronous solution does not provide enough QoS guarantees.

The NAP protocol uses the SpaceWire link layer to arbitrate between different data flows at the beginning of each timeslot.

6.2.2.1 Concept

The key idea is to use timeslots and the wormhole switching mechanism in the SpaceWire routers to arbitrate between data flows, instead of using the global arbiter node described in section 4.2.6.2.

With the NAP protocol, each timeslot, packets from higher priority data flows are sent before the lower priority ones. When the lower priority packets are sent they can become blocked if they use the same network resource, i.e. SpaceWire link, used before by a higher priority one.

Once the packets that did not become blocked finish being transmitted, the blocked packets have a chance to be sent. However, if the next timeslot starts while a packet is still being transmitted it is immediately truncated and discarded by the destination. Figure 6-21 shows the succession of events described for a simple case with two data flows.

Given a group of competing data flows that share the same links or network resources, the one with the highest priority has a guaranteed maximum latency equal to the duration

of a timeslot. For some traffic and topologies this improves the value obtained with asynchronous or scheduled networks.

NAP provides a very efficient distributed TDM with network arbitration using timeslots, SpaceWire routers and the wormhole switching mechanism.

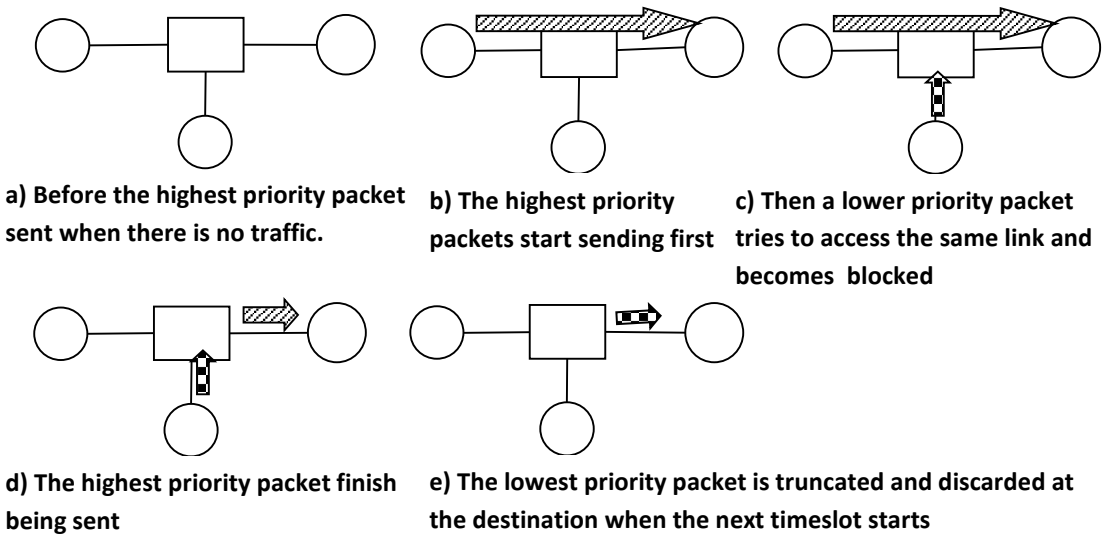


Figure 6-21: Network Arbitration Protocol concept

6.2.2.2 Protocol Specification

The NAP protocol does not specify a packet format but instead it requires the following rules to be followed:

- Each unidirectional data flow defined by a source-destination pair has a network priority level (P) assigned.
- The maximum number of competing data flows that share one or more network resources, i.e. use the same links in the path to the destination, define the number of network priority levels required.
- The network priority level of a data flow can be changed at any time if needed.

- When a timeslot starts the following procedure shall be followed by each node:
    1. Select the data flow with the highest network priority with data pending to be sent.
    2. The selected data flow triggers the sending of one packet after *Tw+Tp*P* time has elapsed since the start of the timeslot, where *Tw* is a constant and *Tp* is the waiting time between each priority level. Note that *P* equals zero provides the highest priority.
    3. When the next timeslot starts, if a packet is still being sent, the source shall cancel and truncate the packet by sending immediately an EEP. The packet should be marked as failed and resent in the next timeslot.
    4. The receiver should discard packets ending with an EEP character that arrive before Tw time has elapsed since the start of the current timeslot.

Figure 6-22 shows how the NAP protocol works when there are three competing data flows from different source nodes that each have a pending packet to send to the same destination. In the first timeslot the high priority packet is sent and the other two become blocked and discarded at the beginning of the next timeslot. In this second timeslot the medium priority packet is sent and the low priority one is discarded.

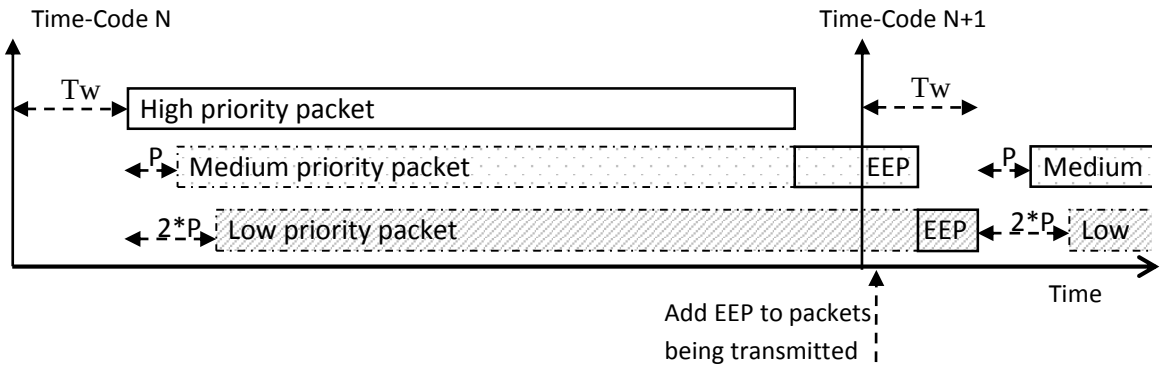


Figure 6-22: NAP protocol operation example

The NAP protocol can be used together with another transport protocol like BTP and LBP. Specifically, the combination of the NAP and LBP protocol is especially efficient as it allows a complete high priority message to be sent and acknowledged in a single timeslot.

6.2.2.3 Protocol Evaluation

The NAP protocol is evaluated in this section without the use of an experimental apparatus. It is considered that a theoretical analysis is enough to evaluate its QoS performance and that the development of an experimental apparatus is too costly for the specific use case of this protocol.

It has been stated that the maximum latency is only one timeslot for the data flow with the highest network priority level. However the duration of a timeslot also determines the protocol efficiency and the maximum throughput that can be achieved.

The minimum duration of a timeslot must be higher than the total arbitration time, which is defined here as the waiting time for the network to discard blocked packets T_W, plus the time until the lowest priority packets start being sent ($N_p * T_p$). Equation (6.5) shows how it is computed and Table 6-14 shows the estimated values for each parameter.

$$T_{arb} = T_W + N_p * T_p$$
$$= N_p * (N_h * L_h) + N_p * \left(\frac{N_h * B * 10}{S}\right) \quad (6.5)$$

Table 6-14: Estimated timing parameters for NAP

Parameter	Value
Maximum number of hops to the destination (N_h)	3
Packet routing latency (L_h)	1 μs
Router total buffer capacity (B)	37 bytes
Link Speed (S)	200Mbps

The number of network priorities N_p is usually set to the maximum number of competing flows, so a different network priority is assigned to each one. Competing flows are data flows from different source nodes that share one or more links in the path to the destination.

Figure 6-23 uses the previous equation to compute the arbitration time depending on the number of competing data flows and the size of the network.

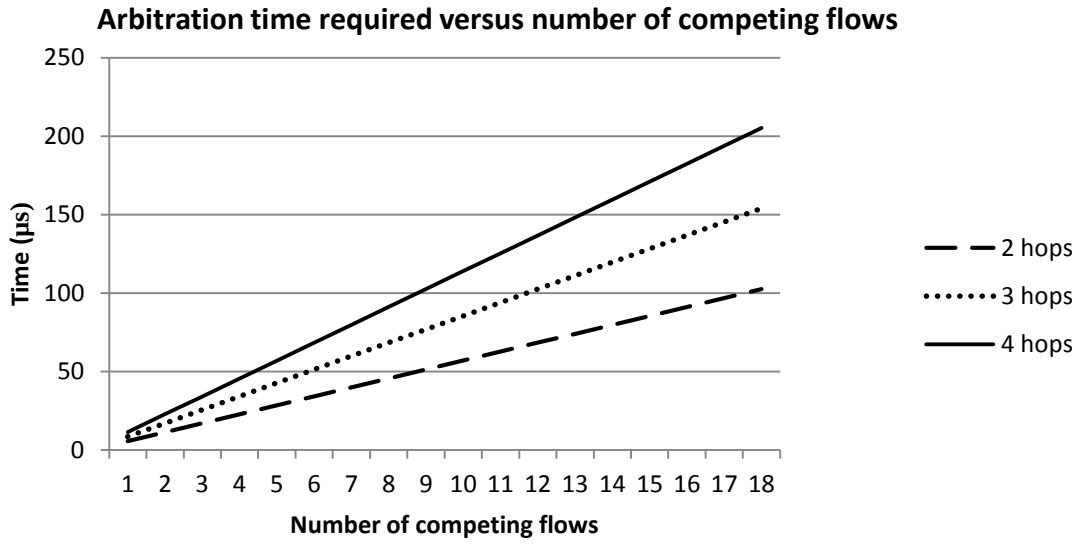


Figure 6-23: NAP arbitration time required versus number of competing flows

The maximum protocol efficiency depending on the timeslot duration can be estimated by assuming that the remaining time from the end of the arbitration until the next timeslot is used to send user data. Figure 6-24 can be used to select a suitable timeslot period depending on the number of competing data flows, the throughput and latency required.

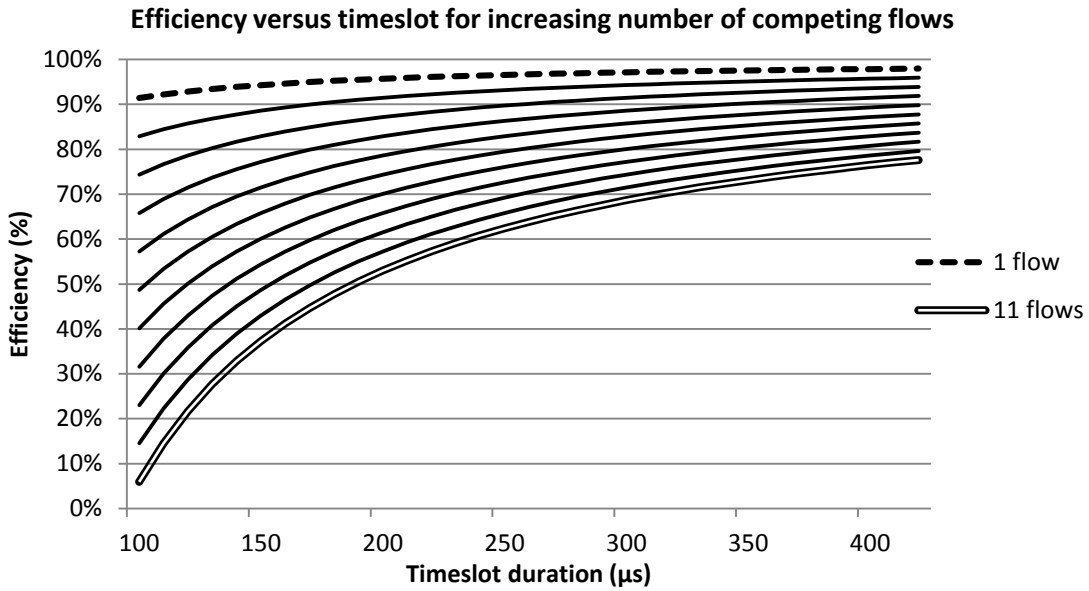


Figure 6-24: NAP efficiency vs slot size for increasing number of competing flows

The most significant issue with the NAP protocol is the bandwidth wasted by packets that are blocked by higher priority packets and are discarded. This makes the protocol inefficient for most typical avionics network topologies, however, there are cases where NAP can be useful.

Figure 6-25 shows a network scenario with a topology and traffic where NAP provides the best QoS performance. There are two data flows that send long packets of 4 Kbytes at 150Mbps raw data rate and a third data flow that sends sporadically small control packets of 256 bytes with only 1Mbps raw data rate.

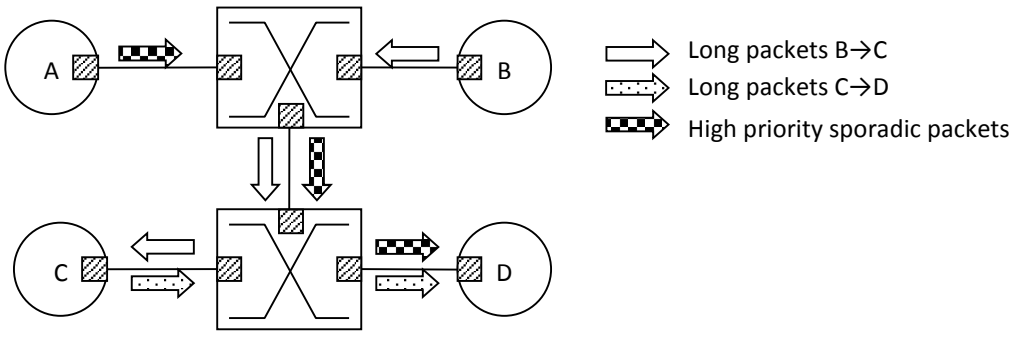


Figure 6-25: Use case for NAP protocol

If the network is scheduled and the link speed is set to 200Mbps, it is required to allocate 3/4 of the timeslots to the high rate data flows. This means that the high priority data flow can only receive one timeslot every four so its worst case latency is 4 timeslots.

If the network is asynchronous the worst case latency is defined by the time it takes to transmit the two competing data flows of 4 Kbytes. This is equivalent to two timeslots, as the minimum timeslot duration is set to the transmission time of 4 Kbytes, around 210µs.

For the NAP protocol the latency is only one timeslot for the high priority packet. Only two network priority levels are required as the two high data-rate data flows do not share any link. The arbitration time in this case is only 17 µs so the timeslot duration can be set to 230 µs, achieving a protocol efficiency of more than 90%.

6.2.3 Summary

This section has presented two protocols, the Link Backpressure Protocol (LBP) and the Network Arbitration Protocol (NAP). They both make use of the cross-layer feedback technique with the SpaceWire link-layer flow control mechanism.

The LBP protocol provides an end-to-end acknowledgement mechanism without requiring the receiver to send an acknowledgement packet. Instead, the packet is acknowledged when the receiver stops receiving data in the middle of the packet reception. This event is detected by the sender thanks to the backpressure effect produced when no SpaceWire FCT characters are being received. The same mechanism can be used to provide basic end-to-end flow control.

One of main benefits of the LBP protocol for asynchronous networks is the small buffer required at the sender side, as each packet is acknowledged immediately, so the maximum

data rate can be achieved with the smallest sending window. For synchronous networks it allows not to have to spend timeslots or reserved bandwidth for the acknowledgements. In any case, with LBP there is no need to define a timeout for the maximum time that the sender must wait to receive an acknowledgement. Its main drawback is the complexity of a software implementation and its low efficiency with small packets (256 bytes or less).

The NAP protocol uses the SpaceWire routers to arbitrate between different data flows from different source-destination pairs that compete for the same resources (links) in a synchronous network. At the beginning of each timeslot, higher priority data flows are sent first, so data flows with lower priority cannot access later the links or network resources used by the higher priority ones. This network arbitration mechanism can provide lower latency to high priority data flows than with scheduled protocols when the traffic is unpredictable or highly not periodic.

In particular, with the NAP protocol, the latency of the highest priority data flow is only the duration of one timeslot, while in case of scheduled networks it depends on the assignment of timeslots. In a scheduled network the latency is only one timeslot duration if all timeslots are assigned to the highest priority data flow which it is usually not the case.

Another advantage of the NAP protocol is that it implements a synchronous network with QoS guarantees that it is simpler to set up, as it does not require to store a schedule table in each node. These QoS guarantees are only related to timely delivery, so reliability must be provided using another transport protocol such as BTP or LBP which defines a specific packet format.

In summary, the NAP protocol is especially suited to network traffic that generates high priority data flows that need to randomly target different destination nodes, but it is not as efficient as scheduled networks when the traffic is predictable and periodic.

6.3 Conclusions

This chapter has shown how cross-layer techniques can be successfully applied to SpaceWire networks to improve the efficiency of the implementation of Quality of Service techniques.

Cross-layer integration is used to develop a protocol optimised for the RMAP protocol. The first RMAP scheduler prototyped in a space-qualified processor has been developed and evaluated. Table 6-9 shows the higher efficiency achieved by the RMAP scheduler protocol in comparison with UTP and BTP when RMAP packets are sent in a synchronous network.

Cross-layer feedback is used in two novel protocols that take advantage of link-layer status information from the SpaceWire link layer. The Link Backpressure Protocol (LBP) is a novel idea based on the utilization of the link-layer status to provide reliability to SpaceWire networks without requiring the sending of acknowledgement packets. The Network Arbitration Protocol (NAP) is based on another novel idea that uses link-layer status information to provide latency guarantees to high-priority data flows in an asynchronous network. These protocols are shown to be most suitable when the network is small and generates a traffic that is highly sporadic or random.

Chapter 7: Link Layer Quality of Service for SpaceFibre

SpaceFibre (SpFi) is a very high-speed full-duplex serial link designed specifically for use onboard spacecraft. The aim of SpaceFibre is to provide point-to-point and networked interconnections for gigabit-rate instruments, mass-memory units, processors and other equipment, onboard spacecraft.

In previous chapters, it has been analysed how to provide QoS to SpaceWire networks using pure or cross-layer transport protocols. In contrast, the development of a new link-layer protocol such as SpaceFibre, is a great opportunity to design and implement QoS capabilities integrated within a link-layer protocol for space applications. Draft A [Parkes 2007] of the SpaceFibre standard (see section 2.1.3) defined the use of virtual channels with independent link-layer flow control. Draft B [Parkes 2010b] outlined the QoS provided by SpaceFibre with the use of FDIR techniques and a Medium Access Controller that arbitrates between each virtual channel.

This chapter will examine the QoS requirements for the new SpaceFibre link-layer protocol and will study of some key QoS implementation aspects. The decisions made are evaluated with the help of the SpaceFibre codec hardware implementation done by the Space Technology Centre with the collaboration of STAR-Dundee and the author of this thesis [Ferrer 2013].

7.1 Overview

This section presents an overview of the main SpaceFibre concepts required for the understanding of the analysis of its QoS aspects. The SpaceFibre standard document [Parkes 2010b] presents for a more complete description.

SpaceFibre is designed to be compatible with the SpaceWire protocol at packet level but providing a much higher data rate. It has multiple channels, called virtual channels (VC), each one with a defined QoS. SpaceFibre has two types of user interfaces to send data:

1. The virtual channel interface comprises a number of virtual channel buffers for sending SpaceWire packets and the same number for receiving SpaceWire packets.
2. The broadcast interface is designed to send short messages of up to 8 bytes with very low latency requirements across the network, in a similar manner as the SpaceWire Time-Codes but providing not only timing distribution but also signalling and interrupt services.

Figure 7-1 shows some SpaceFibre building blocks related with virtual channels and broadcast. The Medium Access Controller selects between each user data interface. The data from virtual channels is segmented into data frames that are sent over the SpaceFibre Link.

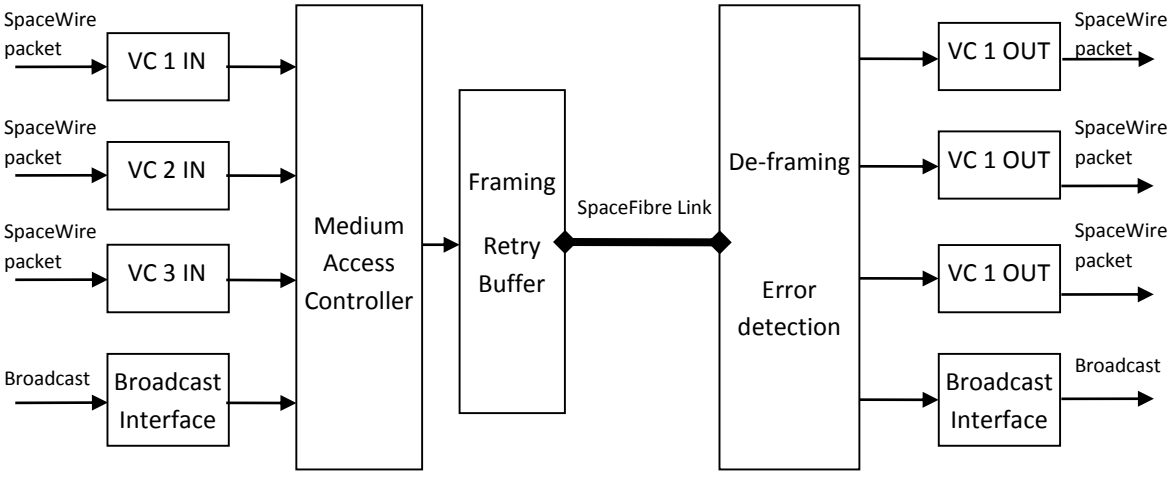


Figure 7-1: SpaceFibre virtual channel and broadcast interfaces

SpaceFibre uses 8b10b encoding [Widmer 1983] which defines symbols of 10 bits that correspond to codes of 9 bits that can be D-codes with one byte of user data or K-codes

used for protocol control. A word is a sequence of four consecutive codes. Data words are four D-codes that contain four bytes of user data. Control words are created with one K-code followed by three D-codes and are used to exchange protocol information.

Frames are blocks of data words that are identified using start and end of frame control words. There are data frames, broadcast frames and idle frames. Data frames can contain up to 256 bytes of user data from a virtual channel. Broadcast frames contain a single broadcast data unit and idle frames are sent when there is no other frame to be sent. Link-layer flow control information is sent using a control word called FCT.

When frames or control words are received they are identified and checked for errors using increasing sequence numbers and CRC codes. If there is an error, a specific control word is sent, called NACK, providing the sequence number of the last valid FCT control word or data frame or broadcast frame. The Retry buffer of Figure 7-1 is responsible for resending data frames, broadcast frames or FCT control words received with errors. They are stored until they are acknowledged with an ACK control word.

Note that control words can be embedded within a data frame as shown in Figure 7-2.

0	8	16	24 31	
COMMA	**SDF**	**VC**	**RESERVED**	Start of Data Frame (SDF) control word
Byte 0	Byte 1	Byte 2	Byte 3	Data frame word 1
Byte 4	Byte 5	Byte 6	Byte 7	Data frame word 2
...	...	...	...	.....
FCT	**Channel#**	**FR_SEQ#**	**CRC**	FCT control word
Byte 60	Byte 61	Byte 62	Byte 63	Data frame word 64
EDF	**FR_SEQ#**	**CRC_LS**	**CRC_MS**	End of Data Frame (EDF) control word

Figure 7-2: SpaceFibre data frame with embedded control words.

7.2 Requirements

The requirements of the SpaceFibre protocol related with Quality of Service are derived from the Draft B of the standard [Parkes 2010b] with inputs from the SpaceWire Working Group [Nomachi 2010].

The SpaceFibre shall provide a completely reliably link with the fastest possible error recovery time for transient and persistent errors using the following FDIR mechanisms:

- Notification of data or control information using positive and negative acknowledgements (ACKS/ NACKS).
- Error detection using sequence numbers, 8b10b error detection capabilities and CRC codes.
- Automatic resending of data frames, broadcast frames and FTCs using a Go-Back-N scheme when sporadic errors occur.
- Automatic reconnection of the link when the error is persistent.

The SpaceFibre shall provide timely delivery and determinism using a Medium Access Controller which arbitrates between virtual channels with the following QoS:

- Priority: provides less latency to virtual channels with higher priority.
- Bandwidth allocation: provides a minimum guaranteed throughput.
- Scheduling: provides deterministic message delivery and minimum jitter.

These different QoS parameters should work together in a consistent manner so it is possible to work at the same time with a virtual channel which requires minimum latency for command and control operations, a virtual channel with a guaranteed throughput for payload data, and a deterministic delivery for minimum jitter or messages that need to be sent and processed in a specific order.

7.3 Design Considerations

This section describes the analysis performed during the development of SpaceFibre regarding its full QoS capabilities.

7.3.1 FDIR

Fault Detection Isolation and Recovery (FDIR) techniques provide the required reliability to SpaceFibre links, which run at much higher data rate than SpaceWire links, so they are more likely to have errors given the same Bit Error Rate (BER). For example, if a BER of 10^{-12} is assumed, which is a reasonable value for optical fibre, seven errors per hour are expected for a single link at 2.5 Gbit/s.

An end-to-end error recovery mechanism is much slower than one implemented at the link layer. It is usually software based, and can take even milliseconds to proceed to resend the data lost, which at gigabit data rates means that the data to be resent needs to be stored in a large buffer. Therefore, it is much better to detect and recover from these continuous errors immediately at link layer by hardware means without waiting for an end-to-end recovery mechanism to take action.

7.3.1.1 Error Detection

During nominal operation, it can be assumed that errors will be produced by a relatively constant expected BER related with Gaussian noise, causing a single bit-flip, i.e. a single bit changes from zero to one or from one to zero in one 10-bit symbol.

During non-nominal operation, an error burst can occur due to interference or some unexpected event, causing multiple bit-flips, i.e. multiple bits that are very close in time are arbitrarily modified.

The objective is to have a link-layer protocol that supports any single bit-flip error and can detect, with a very high probability, multiple bit-flips, so that if there is a sequence of continuous single bit-flips (i.e. BER not nominal) or burst errors, the link is restarted. When the link starts it should check that the BER is nominal before starting to send user data. This guarantees that the link is extremely reliable.

To achieve the objective described the following events needs to be detected or avoided:

- A bit-flip modifies data values in a data frame, broadcast frame or control word.
- A bit-flip converts a data word into a control word, corrupting the protocol operation.
- A bit-flip converts a control word into a data word, corrupting the protocol operation.
- A bit-flip converts a control word into another control word, corrupting the protocol.
- A multiple bit-flip or burst error occurs which must be detected and identified as not being caused by a single bit-flip.
- The BER is non-nominal and too high, causing single bit-flips with high probability.

To deal with the detection of these events, SpaceFibre provides three error detection mechanisms:

1. 8b10b encoding error detection: When a single bit-flip occurs, the disparity of an 8b10b symbol changes. If the disparity of the symbol is non-zero, the error is immediately detected, but if it is zero, then it is not detected until the next symbol with non-zero disparity is received.

2. 8-bit CRC: It is inserted at the end of a control word (see Figure 7-2) or at the end of a broadcast frame.
3. 16-bit CRC: It is inserted at the end of a data frame (see Figure 7-2).

It is possible to determine the values of the control words so that they always have at least one symbol with non-zero disparity. This ensures that when a control word is received, any change in the disparity caused previously by a single bit-flip, is immediately detected.

The SpaceFibre protocol ensures that a control word is always sent after a data frame. This means that if there is a single bit-flip in a data frame, it can be detected in the next word following the data frame, and the whole data frame can then be discarded. If the 16-bit CRC is computed after the next word is received, then it can only be invalid if multiple bit-flips have occurred. Single bit-flips are always detected before by the 8b10b encoding when checking the disparity of the control word following the end of data frame.

Therefore, with this mechanism the 16-bit CRC can be used to monitor when the BER is non-nominal and multi bit-flips or burst errors occur. Note that broadcast frames and other control words are not always followed by another control word, so an error detected with the 8-bit CRC can be due to both single and multiple bit-flip errors.

In theory it is also possible that with a nominal BER, two or more single bit-flips occur within a data frame, which is an error not guaranteed to be detected by the 8b10b decoding. The cumulative distribution function of a binomial distribution can be used to compute the probability that two or more single bit-flips occur in the same frame, as shown in equation (7.1), where n is the number of bits in a frame ($66 \text{ words} * 40 \text{ bits in a word}$ in case of a data frame) and p is the BER.

$$\Pr(K > 2) = 1 - \Pr(K \leq 2)$$
$$= 1 - \sum_{k=0}^{1} \binom{n}{k} p^k (1-p)^{n-k} \quad (7.1)$$

For a BER of 10^{-12} the probability of such an event is as low as $3.5 * 10^{-18}$, so if multiple 16-bit CRC errors occur in a short period (less than a second) it means that the BER is not nominal.

However the probability that two single bit-flips occur in the same frame is not zero and there exists the probability that this error will not be detected by the 8b10b encoding and the CRC, leading to data corruption or protocol failure.

Regarding the CRC, it is important to note that the BER and single bit-flips apply to the stream of 10-bit symbols. The 8b10b encoding implies that a single bit-flip can produce up to six bit modifications in the decoded symbol or D/K Code. The CRC is computed over the 8-bit stream of D-codes and K-codes, and can always detect a single bit-flip in a symbol because it can detect as many consecutive bit changes as the length of the CRC. However, for two single bit-flips in a frame or any other multiple bit-flip error, the worst case probability that the *n*-bit CRC does not detect an error is $1/\ 2^n$.

Regarding the 8b10b encoding, the use of a data scrambler by SpaceFibre implies that the data is random and approximately half of the symbols will have disparity. This greatly increases the chance of an error being detected by the 8b10b encoding before the end of a frame. The actual probability of not detecting an error $P(F)$, can be obtained using the law of total probability for all possible distances between the two bit-flips. Equation (7.2) shows how it is computed, where N is the number of symbols in a frame.

$$P(F) = \sum_{d} P(F|d) * P(d)$$
$$= \sum_{d=1}^{N-1} \left[P_{bE} * (P_{disp} * d) * (P_{bE} * P_{dE})\right] * \left[\frac{N-d}{\binom{N}{2}}\right] \quad (7.2)$$

The first term is the probability that an error is not detected when the bit-flips are separated by d symbols. This is the probability that the first bit-flip does not immediately produce an error (P_{bE}), multiplied by the probability that the following symbols have disparity zero ($P_{disp} * d$), multiplied by the probability that the second bit-flip does not produce an error and the disparity is restored to the original value ($P_{bE} * P_{dE}$). The second term is obtained using the classical definition of probability, i.e. number of cases favourable for the event over the number of total outcomes possible in an equally probable sample space.

Table 7-1: Probabilities related with the 8b10b encoding

Parameter	Value
Probability a bit-flip does not produce immediately an error (P_{bE})	0.31
Probability symbol has zero disparity (P_{disp})	0.5
Probability new disparity is equal to original disparity (P_{dE})	0.5

Figure 7-3 shows the probability of a data corruption per bit received for the data and broadcast frames. It takes into account all error detection capabilities, using equations (7.1, 7.2) with the values of Table 7-1. The expected number of failures can be obtained by multiplying the probability by the number of bits to be sent. As the broadcast frames usually use a very small portion of the bandwidth, both data and broadcast frames offer the same level of reliability, which is extremely high.

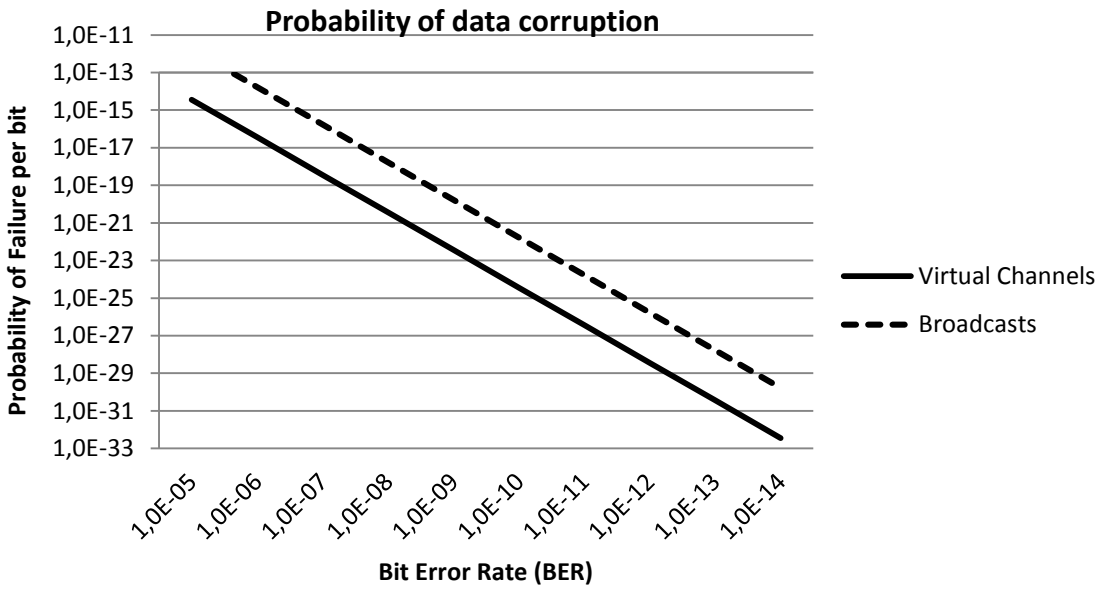


Figure 7-3: SpaceFibre error detection

It has been shown that control words and data frames are protected, but there is also the possibility that a data word with an arbitrary user data value is converted into a control word by a single bit-flip. In this case, the CRC is invalid and it cannot be guaranteed that a symbol with disparity will appear before the control word is processed. However, control words starts with a K-code, so this issue can be avoided by selecting K-codes that cannot be converted into a D-code. As there are K-Codes available that cannot also be converted into a data word by a single bit-flip, this solution also covers the possibility of a control word being converted into a data word.

The possibility of a control word being converted into another control word can be easily covered by the CRC, the protocol rules and specially, by the maximization of the hamming distance between the D-codes used to identify each control word.

7.3.1.2 Error Recovery

Once an error is detected the main requirement for SpaceFibre is to try to recover as fast as possible. This is done by requesting the data sender logic to resend data starting from

the last valid data received. There are two ways to trigger the data receiver logic to resend data:

1. The sender does not receive the acknowledgement (ACK) of the last data sent before a timeout timer expires.
2. The receiver actively sends a negative acknowledgment (NACK) when an error occurs.

The first option is the one implemented in the protocols specified in previous chapters (UTP, BTP, RMAP scheduler and LBP). A main reason was that in the second option, the NACK can also be lost, and this error cause requires the same timeout timer needed by option one to detect that an ACK has not been received. The additional complexity of the NACK indication does not provide a significant benefit.

There is still the possibility that option two can be implemented without a timeout timer but this requires two important rules to be followed:

1. When an error occurs, the receiver needs to send a NACK repeatedly over time, to cover the situation when a NACK is lost.
2. The sender needs to periodically notify to the receiver the sequence number of the last data packet sent (data frame in SpaceFibre). This covers the case when a data packet is lost before arriving at the receiver.

These rules require a lot of unnecessary network bandwidth so it is only reasonable to implement them in a point-to-point connection, which it is only guaranteed to be for a link-layer protocol. Therefore, SpaceFibre can select the option two for its retry or error recovery layer. It has the significant advantage that it does not require to setting of a specific timeout timer, so it can work independently of the speed of the link. More

importantly, the error recovery time does not depend on a timeout timer set but only on the transmission time and the internal logic delay.

For the specific implementation of SpaceFibre the following control words are defined:

- SDF: Start of Data Frame indication
- EDF(#SEQ): End of data frame indication with its sequence number.
- ACK(#SEQ): Acknowledgement with the sequence number of the last data frame received correctly. The sender should free any space used by already-acknowledged data frames.
- NACK(#SEQ): Negative acknowledgment with the sequence of the last data frame received correctly. The sender should resend all data frames sent after the data frame with the sequence number provided.
- SIF(#SEQ): Start of Idle Frame with the sequence of the last data frame sent.

To better analyse and check the details of the operation of the retry layer of SpaceFibre, a software protocol simulator was developed with the following characteristics:

- Hardware ready: Protocol operations are simple and are performed step by step or (clock by clock) so they could be easily translated in hardware when required.
- Extensive error injection test: The user can define a list of scenarios with a specified set of probabilities ranges for the sender data rate, sink data rate, packet length, broadcast, and error injection rate.
- Easy protocol debugging: The simulation can be run with random scenarios with error injection over a long period. When a problem occurs, the last words sent over the link are written in a text file as shown in Figure 7-4.

```
A -> B : DATA [16, 136, 50, 42]                    B -> A :
A -> B : DATA [231, 7, 108, 32]                    B -> A :
A -> B : DATA [140, 84, 32, 162]                   B -> A :
A -> B : DATA [143, 56, 92, 255]                   B -> A :
A -> B : DATA [181, 85, 109, 114]                  B -> A :
A -> B : * DATA [46, 113, 123, 124]                B -> A :
A -> B : DATA [162, 'EOP', 19, 145]                B -> A :
A -> B : DATA [250, 'EOP', 162, 111]               B -> A :
A -> B : DATA [5, 50, 6, 248]                      B -> A :
A -> B : ACK [16, 16]                              B -> A :
A -> B : EDF(24) [0]                               B -> A :
A -> B : SDF [0, 64.0]                             B -> A :
A -> B : DATA [191, 145, 212, 5]                   B -> A : NACK [22, 22]
A -> B : DATA [230, 139, 250, 94]                  B -> A : SIF(16)
A -> B : DATA [38, 101, 76, 156]                   B -> A :
A -> B : DATA [220, 93, 186, 45]                   B -> A :
A -> B : DATA [151, 150, 113, 251]                 B -> A :
A -> B : DATA [255, 72, 216, 99]                   B -> A :
A -> B : DATA [35, 74, 186, 98]                    B -> A :
A -> B : DATA [177, 25, 204, 156]                  B -> A :
```

Figure 7-4: SpaceFibre retry layer protocol debugging

One of the most critical issues observed using the simulation tool was related with the sending of multiple NACKs when a single error occurs. Figure 7-5 shows how two consecutive data frames are sent twice when a single error occurs, due to the reception of multiple NACKs. The problem is that the sender cannot know if the second NACK received is due to a second error in the frame resent or to previously sent frames received by the receiver after the first error occurred.

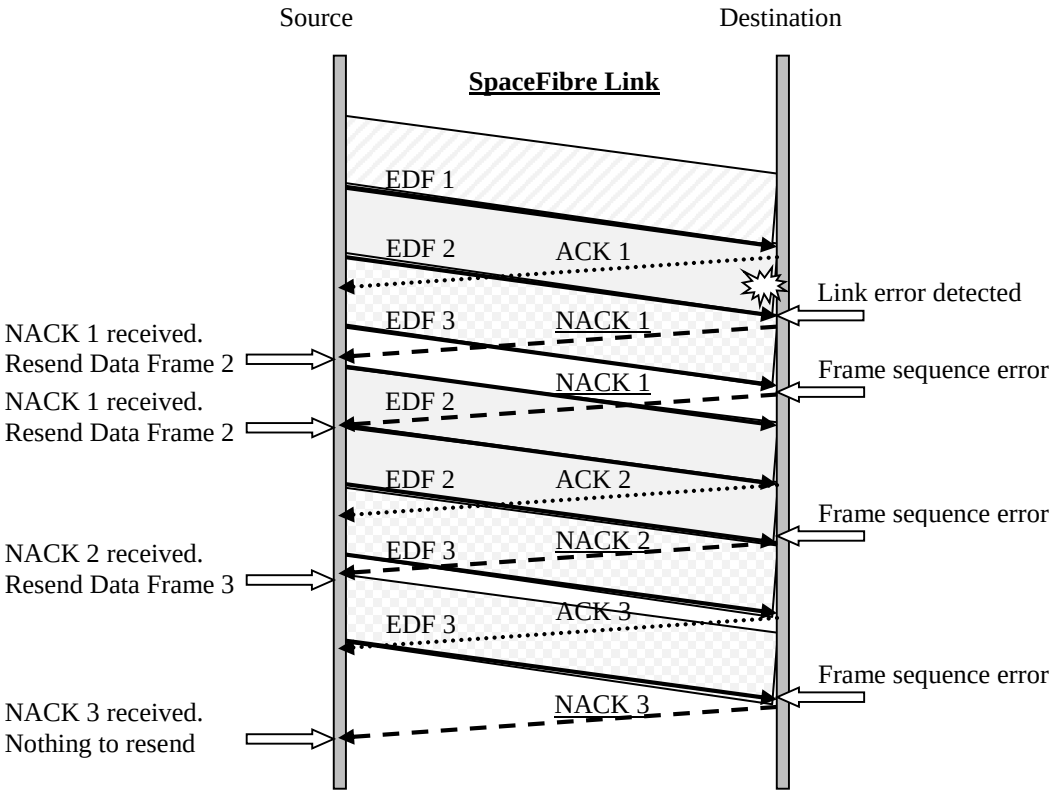


Figure 7-5: Issue with multiple retries when a single link error occurs

The solution proposed by the author of this thesis was to use a 1-bit polarity flag within the 8-bit sequence field. Then the 7-bit sequence number can be positive or negative. The sequence number is still increased by one each time a data frame is sent or correctly received. On the other hand, the sender polarity, the polarity stored in the data sender logic, and the receiver polarity, the polarity stored in the data receiver logic, should follow the rules below:

- If the sender receives a NACK with the same sender polarity, the NACK is processed and the sender polarity is changed. If the NACK has different polarity the NACK is ignored.
- If the receiver detects an error it sends NACKs with the polarity of the last data frame correctly received until a new data frame is correctly received with the modified polarity.

With this mechanism the second NACK of Figure 7-5 is discarded and a data frame is not resent twice when a single link error occurs. Figure 7-6 shows how the proposed mechanism works with the polarity specified, with positive and negative marks in the sequence numbers.

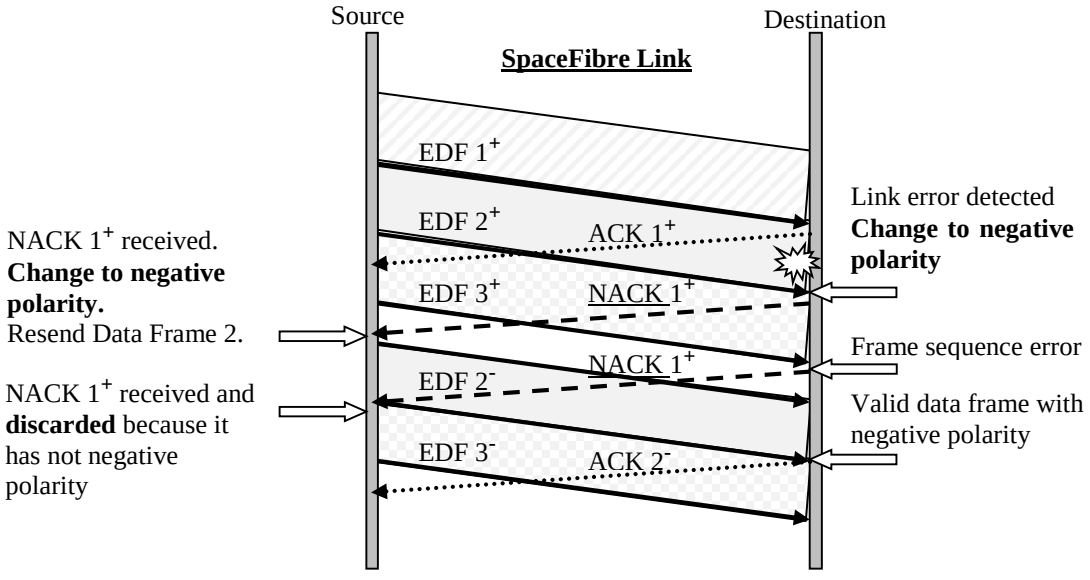


Figure 7-6: SpaceFibre polarity mechanism

The simple rules described do not cover all possible error cases, in particular when two consecutive errors occur, so the actual implementation of the polarity mechanism in the receiver is a little bit more complex. It is described with the state machine shown in Figure 7-7, where each state is defined by the error condition and by the value of the polarity flag. The polarity flag is used to check sequence frames received. Note that when an error occurs, this state machine is updated before a NACK is sent. Also, NACKs are always sent with the polarity flag inverted so the resulting frame sequences of Figure 7-6 are obtained.

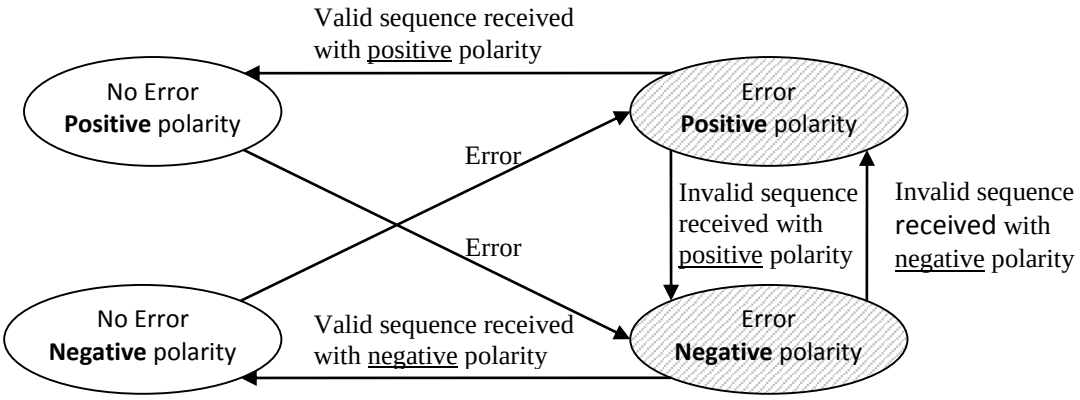


Figure 7-7: Receiver polarity state machine

7.3.2 Medium Access Controller

The SpaceFibre standard defines a Medium Access Controller (MAC) which determines which virtual channels can send data and in which order. The timely delivery and deterministic characteristics of each virtual channel depends on the implementation of the MAC and its current configuration. The Draft B of the SpaceFibre standard defined three different basic QoS mechanisms: bandwidth reservation, priority and scheduling.

7.3.2.1 Bandwidth Reservation

Bandwidth reservation QoS is implemented in the MAC using a bandwidth credit counter for each virtual channel. The bandwidth credit counter monitors the amount of data sent by a virtual channel relative to the amount data that it is allowed to send. This value increases over time by the amount of data that can be sent by the link, and it decreases each time a data frame is sent by the amount of data sent, relative to the expected portion of the bandwidth reserved for this virtual channel.

Equation (7.3) provides the mathematical expression for the evolution over time of the Credit Counter for a virtual channel, $C_i(t)$, where N is the number of virtual channels, S is the link speed, $D_i(t)$ is the amount of Data sent by virtual channel i, $D(t)$ is the total amount of data sent, and B_i is the portion of Bandwidth allocated for this virtual channel, see equation (7.4).

$$C_i(t) = \int \left(D(t) - \frac{D_i(t)}{B/B_i} \right) dt \tag{7.3}$$

$$B = \frac{\int_{t=0}^{t=1s} D(t)dt}{S} \geq \sum_{i=1}^{i=N} B_i \tag{7.4}$$

In SpaceWire Draft B, the Credit Counter directly determines the Precedence of the virtual channel. The MAC selects the virtual channel with the highest precedence. Therefore, if a virtual channel sends less data than expected, its Credit Counter will increase over time and it will be selected by the MAC over other virtual channels with a smaller Credit Counter value. The Credit Counter is negative if it is sending more data than expected and if it is zero it means that virtual channel has used exactly the expected bandwidth portion allocated.

This Credit Counter mechanism was evaluated with a dedicated simulation tool developed in software using Python. Figure 7-8 shows a screenshot of the MAC Virtual Channel (VC) simulator configured to produce the results of Figure 7-9. Note that the VC1 has a constant throughput and the other two VCs send data generated by a Poisson source. That is why the Credit Counter and bandwidth used fluctuates for VC2 and VC3 but it is constant for VC1.

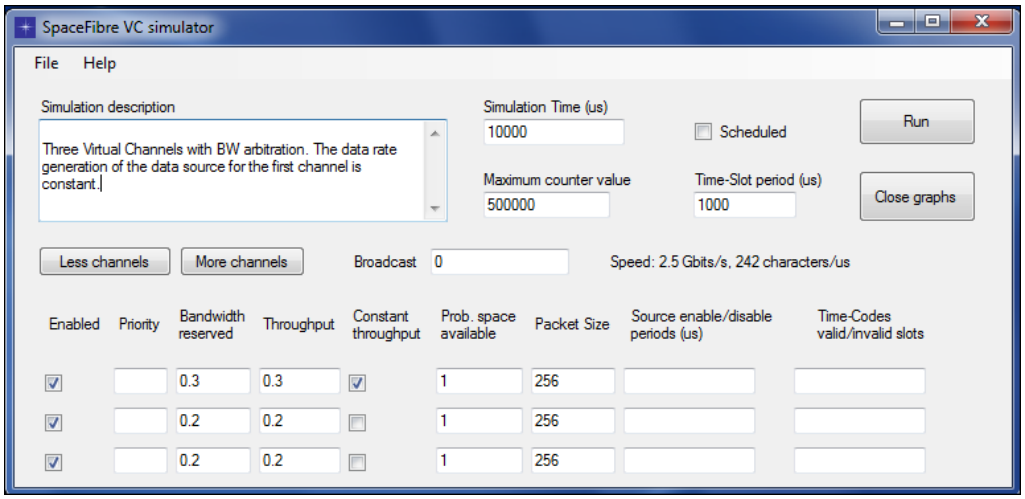


Figure 7-8: SpaceFibre MAC simulator

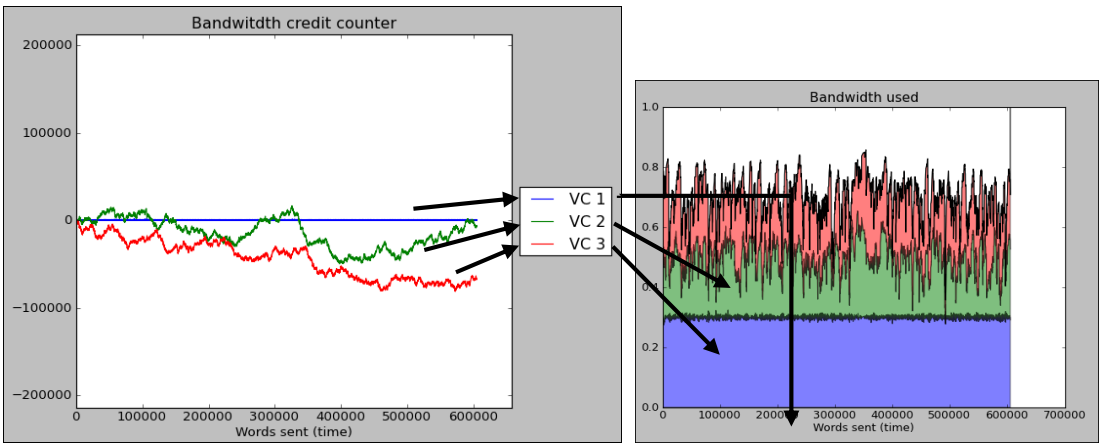


Figure 7-9: Credit Counter and Bandwidth Used by three VCs

7.3.2.2 Priorities

In SpaceFibre draft B, a VC with Priority QoS was determined by setting a fixed precedence value over the range of possible precedence values given by the credit counter. The idea is that a VC with a high priority value should be selected before another VC with Bandwidth Reserved QoS that has sent more data than expected. However, there are two issues with this approach:

1. In a real case, the portion of bandwidth allocated to a virtual channel will be never exactly the same as the bandwidth measured, so any small difference will produce over time the saturation of the Credit Counter value to the maximum or to the minimum value. This means that the fixed priority precedence set within the credit counter range will have little sense.

   Figure 7-10 shows how Credit Counter saturates to the maximum value when the throughput is a little bit lower than the bandwidth allocated. Note that the arbitration mechanism works well giving the same results as Figure 7-9.
2. It is not possible to assign a bandwidth allocated portion to a VC with Priority QoS. Therefore, it is not possible to have two or more high priority virtual channels that are arbitrated by the MAC based on its bandwidth allocated.

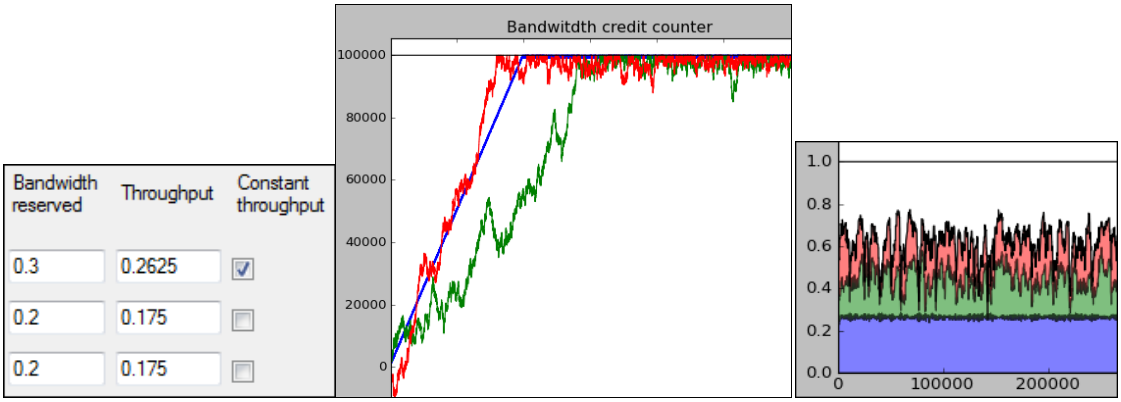


Figure 7-10: Credit Counter saturation

The solution proposed was simply to make this second point possible, by defining as many non-overlapping precedence ranges as Priority levels required. For each precedence range or priority level, the Credit Counter value can fluctuate, as illustrated in Figure 7-11, according to the bandwidth being used. As explained, the MAC selects the channel with the highest precedence with data ready to be sent.

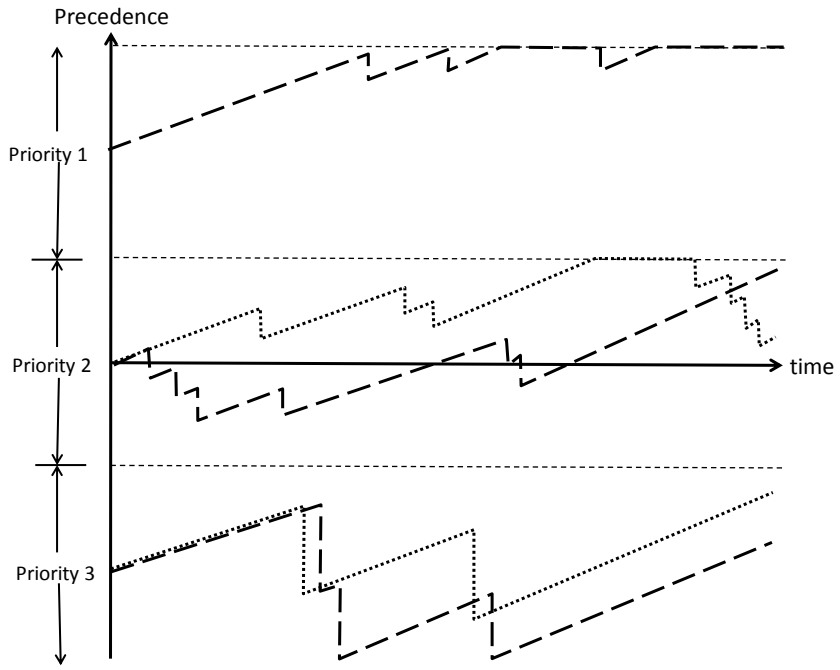


Figure 7-11: Combined Priority and Bandwidth Reservation QoS

This scheme allows very good integration between the Priority and Bandwidth Reservation QoS, using only two parameters assigned to each VC, the priority level and the bandwidth allocated portion.

- The priority level determines the link latency of a VC. Specifically, the link latency of a VC will be close to zero if other VCs with higher priority have no data to be sent.
- The bandwidth allocated portion parameter sets the minimum guaranteed link throughput of the VC.

Figure 7-12 shows the latency of data frames obtained using the simulator for two virtual channels represented by dark and light dots. The horizontal axis shows the simulation time. The latency is measured on number of characters transmitted while a data frame is waiting to be sent. The difference between the left and the right panel is that in the right panel the priority of the virtual channel related with black dot frames have been set higher than the lighter frames, leading to a reduction in their latency. This shows that the priority level of a virtual channel determines the latency.

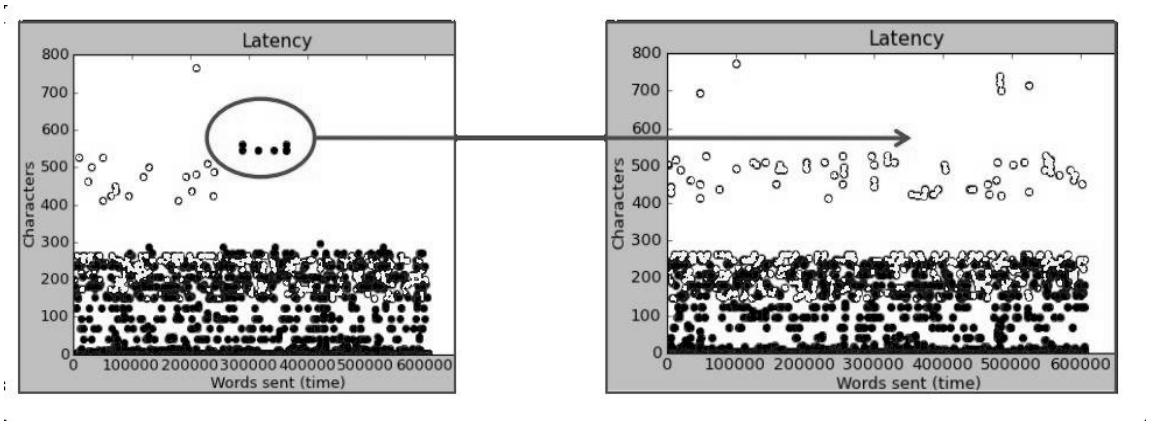


Figure 7-12: Higher priority settings reduces virtual channel latency

Note that the link latency and link throughput should not be confused with the end-to-end latency and throughput evaluated in the previous chapters. SpaceFibre uses virtual channels, so the end-to-end QoS metrics of a data flow can be easily determined if no other data flow is using the same virtual channel in any hop along the path to the destination. The end-to-end throughput is equal to the minimum link throughput guaranteed along the path, and the end-to-end latency is the sum of the link latencies experienced along the path.

Finally it is worth noticing that this scheme allows the detection of babbling idiot errors, which happen when a node starts sending data continuously due to an anomaly. The bandwidth allocated portion is configured so that the actual value is higher than the expected one, causing the upper saturation of all Credit Counters under nominal

conditions. If a babbling idiot error occurs, the related VC will have its Credit Counter saturated at the lowest level of their range, which is set by their priority level. This error condition can be flagged to the system which can decide to disable the VCs that have this non-nominal Credit Counter value.

7.3.2.3 Scheduling

In previous chapters, the scheduling mechanism has been used as a TDM technique that provided guarantees for the latency and the throughput. For this purpose, the timeslot duration had to be trade off to be as small as possible, without decreasing the link utilisation too much due to packet processing overheads.

However, SpaceFibre already provides QoS guarantees in throughput and latency using the bandwidth allocated portion and priority settings. Therefore, a scheduling mechanism with longer timeslots can be used for other purposes, especially for the following two use cases:

1. Scheduling can be used by the user application to provide deterministic in-order delivery for data flows coming from two different sources. Also a destination node may not want to receive data while it is receiving high priority commands or other data from another VC.
2. Scheduling could also be used to provide guarantees for the maximum latency and minimum throughput when two or more source nodes are using the same virtual channel to send data to the same destination node or output router port. The scheduling mechanism can be used to arbitrate between different data flows using the same virtual channel the same way that it was used in previous chapters for SpaceWire to arbitrate between different data flows competing for the same port in a router.

Figure 7-13 shows an example of a network topology that illustrates how the scheduling mechanism can be used to avoid that a node could receive data packets simultaneously from two different virtual channels. A SpaceFibre Routing Switch interconnects multiple SpaceFibre ports, each one with two virtual channels.

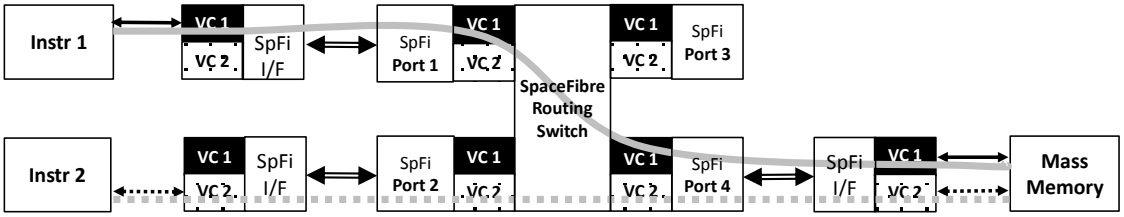


Figure 7-13: SpaceFibre network using scheduling with different virtual channels

The scheduling table is shown in Table 7-2, which indicates that half of the available timeslots, i.e.0-127, are used by the Instrument 1 to send data packets to the Mass Memory using VC 1. Instrument 2 sends during the other half of timeslots data to the Mass Memory using VC 2.

Table 7-2: SpaceFibre Network Scheduling for different virtual channels

	Timeslots 0-127	**Timeslots 128-256**
VC 1	Instrument 1 → Mass Memory	-
VC 2	-	Instrument 2 → Mass Memory

Figure 7-14 shows the network topology when scheduling is used to arbitrate between two flows from different source nodes which use the same virtual channel.

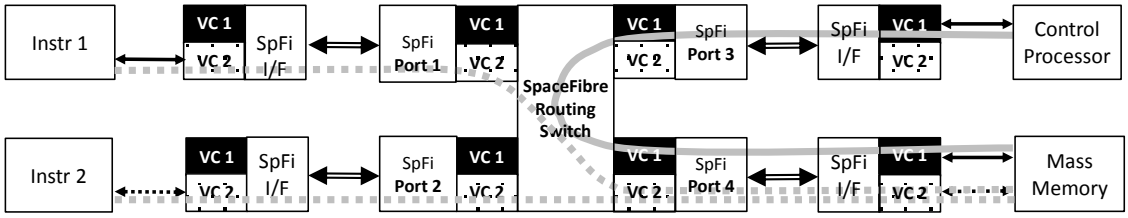


Figure 7-14: SpaceFibre network using scheduling for the user application

Within the SpaceFibre Routing Switch, the data coming from a virtual channel are SpaceWire packets that need to be routed to a specific SpaceFibre port and VC. As expected, the output port selected depends on the SpaceWire path or logical addressing and round robin arbitration is used. The VC selection policy here is as simple as selecting the VC with the same number used by the input port.

The scheduling table for this scenario is shown in Table 7-3, which indicates that during even timeslots the Instrument 1 can send data packets to the Mass Memory using VC 2. During odd timeslots Instrument 2 can send data packets to the Mass Memory using VC 2. Finally, in all timeslots the Control Processor can send data to the Mass Memory using VC1.

Table 7-3: SpaceFibre Network Scheduling for port arbitration

	Timeslots 2*N	**Timeslots 2*N+1**
VC 1	Control Processor → Mass Memory	Control Processor → Mass Memory
VC 2	Instrument 1 → Mass Memory	Instrument 2 → Mass Memory

This scheduling mechanism can be implemented in a similar way as done in previous chapters, using broadcast frames instead of Time-Codes, which allows up to 256 timeslots. However, the scheduling operation is applied frame-by-frame instead of the packet-by-packet operation done with the previous SpaceWire protocols.

Note that for the cases considered, it is an error condition that a packet is being sent when the timeslot changes. The nodes should send data synchronised with the timeslot and not send more data than can be sent in the current timeslot. In case of malfunction and a packet still being sent when the timeslot changes, the SpaceFibre router should add an EEP and spill the rest of the packet. If this EEP is not sent, the router could not free the output port being used and this would lead to error propagation in the next timeslot.

7.4 Experimental Apparatus

Some key design considerations presented in the previous chapter were the basis for the QoS implementation described in the next revisions of the SpaceFibre specification. Draft D included the new Medium Access Controller with multiple bandwidth credit ranges. Draft E specified the FDIR with the retry operation using the polarity mechanism previously described.

The new SpaceFibre specifications were prototyped in hardware, so the evaluation of the QoS mechanisms discussed before became possible. STAR-Dundee in collaboration with the University of Dundee developed STAR Fire [Ferrer 2013], a complete SpaceFibre diagnostic unit. STAR Fire has two independent SpaceFibre interfaces compliant with the latest draft of SpaceFibre standard [Parkes 2012], each one with an embedded link analyser and multiple very high data rate hardware data generators and checkers.

Figure 7-15 shows the STAR Fire unit with two SpaceWire interfaces and the eSATA connectors used by the SpaceFibre interfaces. The SpaceWire interfaces and SpaceFibre virtual channels are connected to an embedded SpaceWire router as shown in Figure 7-16. This allows SpaceWire packets from SpaceWire interfaces to go into in SpaceFibre Virtual channels and vice versa. However, in order to achieve the much higher data rate of SpaceFibre, the hardware data generators and checkers are used.

Figure 7-15: STAR Fire front view

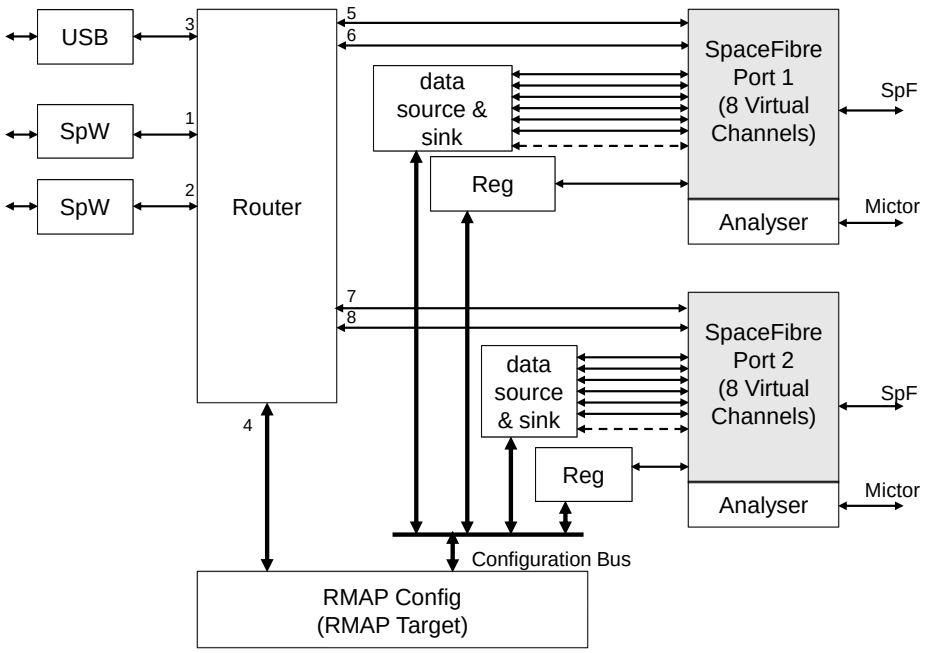


Figure 7-16: STAR Fire block diagram

In order to control and monitor the STAR Fire unit, dedicated software was developed by the author with a Graphical User Interface (GUI). The GUI allowed to easily configure the unit and set up the trigger of the embedded analyser as shown in Figure 7-17.

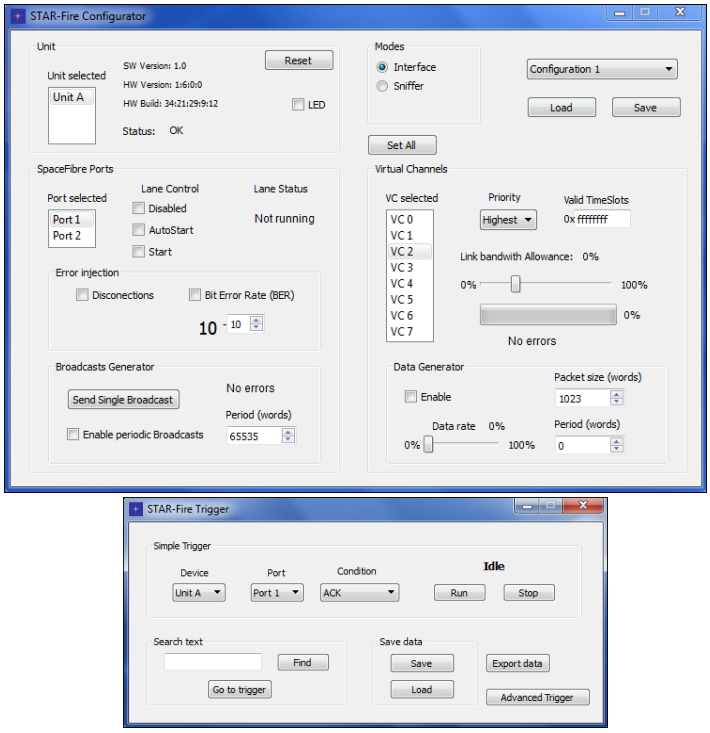


Figure 7-17: Screenshots of the STAR Fire software

7.5 Evaluation

The experimental apparatus described was used to evaluate the link-layer Quality of Service implemented in SpaceFibre with FDIR and MAC mechanisms, related to reliability and timely delivery.

7.5.1 FDIR

The FDIR capabilities of SpaceFibre were tested by injecting errors and checking that the errors were recovered using the link-layer retry mechanism, so the link continued to operate without any data corruption or loss.

Using the GUI previously described, single bit-flip errors detected by the 8b10b encoding were injected and no data errors were detected by the hardware data checkers, as shown in Figure 7-18.

If more than one bit-flip per data frame was injected continuously with the link not being disconnected, data errors were observed after some time. This is expected and described in section 7.3.1.1. In a real case, this would not be a problem because continuous multiple bit-flips per frame lead to a very high BER which is detected and causes the link to be automatically disconnected.

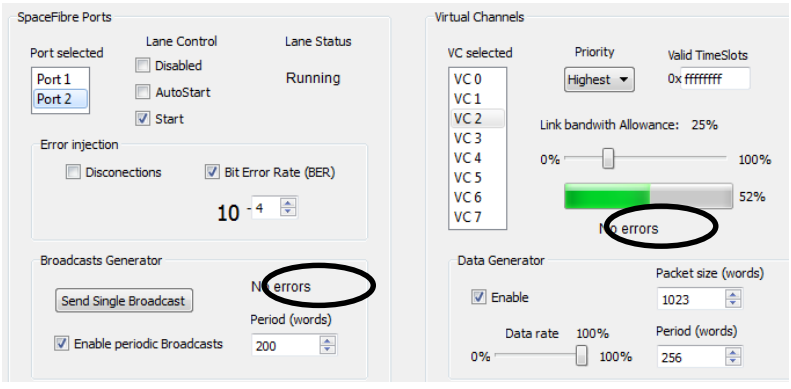


Figure 7-18: Error injection with STAR Fire software

Figure 7-19 shows a retry event captured by the STAR Fire analyser while errors are being injected. After a NACK control word is received, a RETRY control word is sent. It is then followed by the frames sent with the sequence numbers following the NACK sequence number received. Note that the FCT control words are sent before the data frames as they have higher precedence, hence the frames are not resent in the same original order to minimise latency of higher precedence frames.

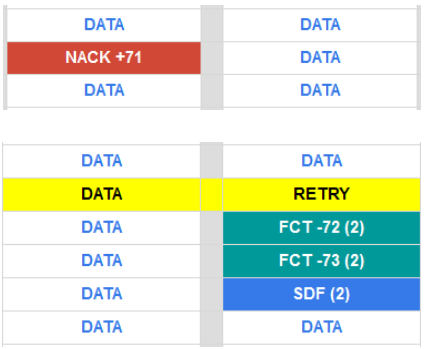


Figure 7-19: SpaceFibre retry event captured by the STAR Fire analyser

The error recovery time or the delay in the reception of data when an error occurs was measured and it was always less than the maximum theoretically expected. For SpaceFibre, using the mechanism defined in section 7.3.1.2, it is defined by the following equation, where D_{frame} is the time required to transmit a frame, D_{logic} is the delay introduced by the hardware implementation, D_{cable} is the transmission cable delay, S is the link speed, L is the length of the cable and $f * c$ is the speed of light within the cable.

$$D_{error} = D_{frame} + 2 * (D_{logic} + D_{cable}) = \frac{66 * 40}{S} + 2 * (D_{logic} + \frac{L}{f * c}) \qquad (7.5)$$

With STAR Fire and a cable of less than a metre, the D_{error} measured was always less than three microseconds. This small value means that the buffers do not need to store a large amount of data when an error occurs in order to keep up with a continuous user-application data-rate source.

This small error recovery time also implies that the throughput does not decrease significantly for nominal BERs. Figure 7-20 shows the values obtained experimentally with STAR Fire. Note that the throughput recorded is with all errors being recovered by retries so that the end user application does not see any errors.

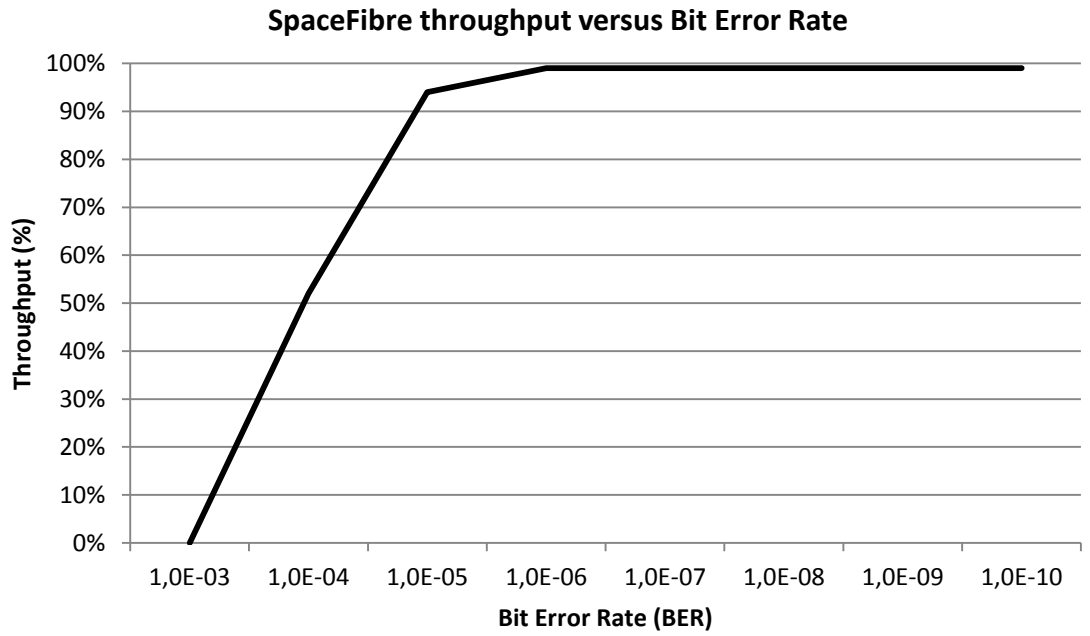


Figure 7-20: SpaceFibre retry event captured by the STAR Fire analyser

7.5.2 Medium Access Controller

The MAC mechanism that allows to combine bandwidth reservation, priorities and scheduling techniques is evaluated with the following scenario:

- One Virtual Channel (VC4) is set with a bandwidth reservation portion of 20%.
- Another Virtual Channel (VC5) is set with a bandwidth reservation portion of 30%.
- Both have the same priority level.
- Both are set to use only half of the total timeslots.

- Both are set to have a data source that tries to send data continuously (100% data rate)

Figure 7-21 shows the settings of this scenario with the STAR Fire configuration tool and the resulting data rate provided by the bar below the slider. Note that VC 4 and VC 5 use the same timeslots and they are alternatively sending and stopping (0xa = 1010b).

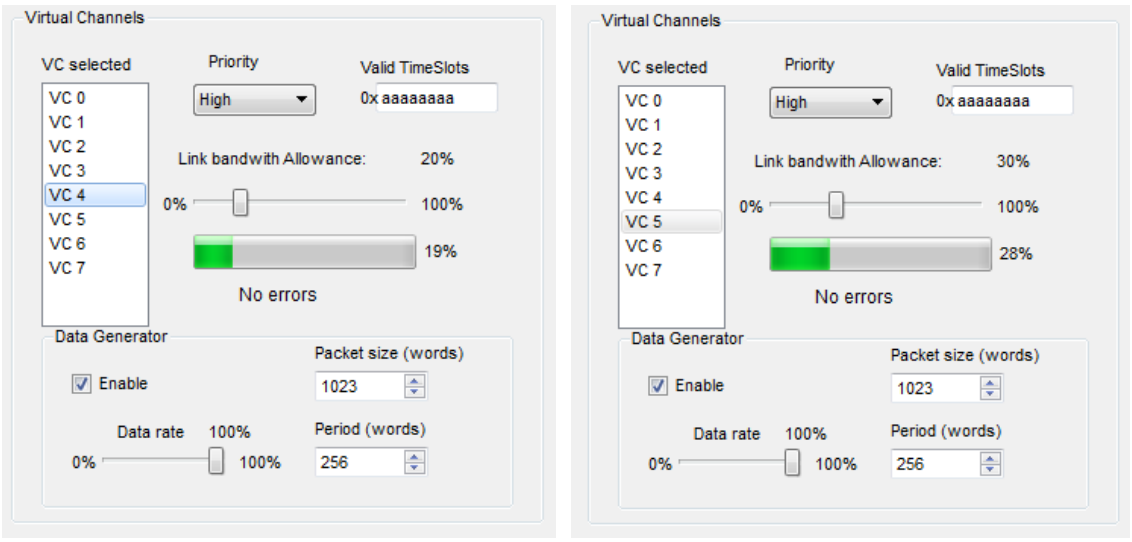


Figure 7-21: STAR Fire configuration with combined QoS mechanisms

As expected, the sum of the data rate of VC4 and VC5 is not more than 50%, as the scheduling mechanism prevents half of the timeslots to be used to send data from these virtual channels. Also, each VC obtains the minimum throughput allocated (~ 20% and ~30%). The source data rate of both VCs was the same but the bandwidth reservation mechanism worked to provide different throughput portions.

The priority mechanism can now be evaluated by adding a third virtual channel that it is allowed to send data in any timeslot (value set to 0xffffffff) with a source that tries sending data continuously.

- If the new virtual channel (VC 2) has a higher priority than VC 4 and VC 5, then it will take all the link bandwidth and so not allow VC 4 and VC 5 to send any data.
- If the new virtual channel has a lower priority than the other two virtual channels, then it only takes half of the total link bandwidth, corresponding to the half of the timeslots that are not being used. Figure 7-22 shows the STAR Fire tool status in this case.

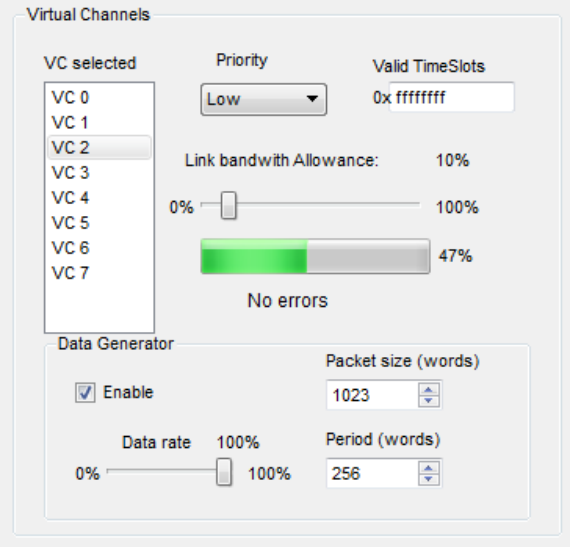


Figure 7-22: STAR Fire configuration with combined QoS mechanisms (2)

This test case described has illustrated how the three QoS mechanism can work together in a consistent way.

7.6 Conclusions

This chapter has reviewed the QoS requirements for the new high-speed SpaceFibre link protocol, analysed how different QoS techniques can be implemented and evaluated them with an experimental apparatus.

Contrary to the protocols developed for SpaceWire in previous chapters, SpaceFibre has offered the opportunity to design QoS mechanisms at the link layer. Also, the use by SpaceFibre of virtual channels has been a critical tool for the deployment of advanced QoS mechanism, like bandwidth reservation. This technique uses a Medium Access Controller to arbitrate between each virtual channel on a frame-by-frame basis, holding in each frame a portion of a SpaceWire packet. Note that with SpaceWire, the bandwidth reservation technique could not be implemented as it is difficult to determine the actual bandwidth used at network level due to the congestion created by other flows. With SpaceFibre, the bandwidth reservation applies only at link layer when no congestion can occur.

Therefore, a suitable Medium Access Controller has been designed and prototyped for SpaceFibre that can simultaneously implement priority, bandwidth allocation and scheduling QoS techniques in a consistent manner.

Regarding the reliability of SpaceFibre, this chapter has presented a FDIR scheme that achieves a mathematically proven extreme probability of error detection and implements the fastest possible error recovery procedure.

The error detection scheme designed has two layers. The first layer uses the 8b10b encoding and the 8-bit CRC to detect any single bit-flip affecting a frame. The second layer uses the 16-bit CRC to detect when the link is being affected by bursts errors. In

addition the Bit Error Rate is constantly monitored and the link is automatically disconnected when the value is too low.

The error recovery procedure relies on the active sending by the receiver of negative acknowledgements when an error occurs. A novel solution based on using frame sequence polarities allows to avoid the setting of a timeout value that in other systems needs to be set based on the link speed.

The author of this thesis developed the software which monitors and controls the experiment apparatus and was also heavily involved in the hardware design and coding of the FDIR capabilities of SpaceFibre.

Chapter 8: Conclusions

This chapter contains the conclusions of the research presented in this thesis. Answers to each of the research questions are provided, the importance of the research is described, the limitations of the research are identified, and future research in this area is suggested. The final section of this chapter contains the conclusions which may be drawn from the research.

8.1 Answers to the Research Questions

Each of the three research questions listed at the start of this thesis have been answered.

8.1.1 Quality of Service for SpaceWire

The first question, "How should a Quality of Service layer be implemented for SpaceWire?", was answered in Chapters 4 and 5. Chapter 4 reviewed the main aspects of the Quality of Service paradigm and how they could be applied to SpaceWire networks. Chapter 5 presented two different transport layer protocols that provide Quality of Service in terms of reliability and timeliness without requiring any modifications to the SpaceWire Standard.

The first protocol designed, the Bidirectional Transport Protocol (BTP), is an all-purpose transport protocol especially suited to networks that use bidirectional data flows. Reliability is provided using acknowledgments, retrying the data when an error occurs. High protocol efficiency is achieved using piggybacking, with the encapsulation of protocol control information and user data in a single packet. Timely delivery is provided using sending priorities and a Time Division Multiplexing technique when the network is scheduled. When the network is not scheduled, latency guarantees can also be provided in certain cases using segmentation and offline network analysis.

The second protocol designed, the Unidirectional Transport Protocol (UTP), is specifically developed for scheduled networks using unidirectional data transfers. The protocol efficiency for this typical use case in space applications is improved by dividing timeslots into three phases. They correspond to the three basic communication steps: notify that receiver is ready, transfer data and acknowledge data received.

The rationale behind the development of these two protocols should be considered when a Quality of Service layer for SpaceWire is designed, in particular:

- Protocol operation complexity and number of SpaceWire packets sent per SDU should be minimised to cope with the limitations of space electronics.
- Segmentation and sending priorities should be applied in scheduled networks to improve the protocol efficiency.
- Timeslot period and scheduling tables should be set to meet the latency and throughput requirements of each data flow, i.e. network metrics, and they should be independent (decoupled) of any user-application scheduling.

8.1.2 Cross – layer Optimizations for SpaceWire

The second question, "Is it useful to use Cross-Layer Quality of Service techniques over SpaceWire?", was answered in Chapter 6. The answer is affirmative because two significant cross-layer techniques used in other application domains were applied successfully in SpaceWire networks.

Cross-layer integration was used to develop a protocol optimised for the RMAP protocol, one of the most used application-layer protocols in SpaceWire networks. Table 6-9 shows the higher efficiency achieved by the RMAP scheduler protocol in comparison with UTP and BTP when RMAP packets are sent in a synchronous network.

Cross-layer feedback was used to apply link-layer status information from the SpaceWire link layer to two new protocols developed. The Link Backpressure Protocol (LBP) uses link-layer status to provide reliability to SpaceWire networks without requiring the sending of acknowledgement packets. The Network Arbitration Protocol (NAP) uses link-layer status information to provide latency guarantees to high-priority data flows in an asynchronous network. These protocols are shown to be most suitable when the network is small and generates a traffic that is highly sporadic or random.

8.1.3 Quality of Service for SpaceFibre

The last question, "How should the new SpaceFibre protocol be designed to provide the required Quality of Service?", was answered in Chapter 6. The starting point was the SpaceFibre specification Draft B [Parkes 2010b] which defined the use of virtual channels and the support of the following Quality of Service types: bandwidth reservation, priority and determinism. After an extensive simulation work, it became clear that it is possible to integrate all these Quality of Service requirements using a single mechanism based on a novel multi-layer priority scheme (see details in section 7.3.2).

The reliability aspects of the initial SpaceFibre specification were also simulated and evaluated. The error detection capabilities of the CRC and the 8b10b encoding are computed analytically and they are used together in a consistent manner to detect the type of the error condition produced (i.e. single bit error, or burst error).

Finally an error recovery mechanism was devised by the author of this thesis that minimises the error recovery time, even when multiple consecutive errors occurs. It is a mechanism that can be applied to any link-layer protocol and it does not use timeout timers. Instead, it is based on the idea of changing the type of sequence numbers each time user data is resent.

8.2 Importance of This Research

The topic of this research, Quality of Service for networks onboard spacecraft, is of high interest for the industry working with spacecraft avionics. The reason is that it can lead to an important cost reduction in the avionics design and validation, especially for complex payload data-handling systems. The Quality of Service paradigm, besides the obvious new network capabilities that it provides, avoids the design of ad-hoc solutions and enables the decoupling of the network from the user application. This simplifies the overall system validation and allows the design of network nodes or clients independently of each other.

In particular, the research performed in this thesis has been an important topic discussed during multiple SpaceWire Working Groups. The author of this thesis has presented different protocols and techniques in these meetings and has participated in a total of fourteen publications. They will hopefully be a source of inspiration for other research teams.

The work presented in this thesis has been a key contribution to the development and prototyping of SpaceFibre and SpaceWire-D. These two protocols are in the process of being standardised by European Cooperation for Space Standardization (ECSS).

8.3 Novelty contributions

A list of the main novelty contributions is presented below in order of being presented in this thesis:

- In depth study of how to apply QoS to SpaceWire networks.

- Simple computation of an upper-bound packet latency for wormhole switching networks.
- Cross-layer integration for wormhole switching networks, in particular for SpaceWire.
- SpaceWire scheduling with sending priorities and optimised use of timeslots.
- First implementation of an RMAP scheduling in FPGA.
- Specification, simulation and evaluation of the retry mechanism of SpaceFibre.
- First simulation and evaluation of the MAC controller of SpaceFibre that supports bandwidth allocation, priorities and scheduling.

All experimental apparatus described in this work have been developed only by the author of this thesis except for two cases. In section in 6.1.5, a trainee helped with the development of part of the VHDL code. In section 7.4 the hardware was produced by a team with members of University of Dundee and STAR-Dundee.

8.4 Limitations

This thesis has investigated the implementation of Quality of Service for SpaceWire networks using existing space-qualified equipment and has not considered the modification of the SpaceWire standard specification.

8.5 Future Work

It is expected that into the near future a new revision of the SpaceWire standard will provide signalling mechanisms using SpaceWire Time-Code characters. This new feature could be a powerful tool to improve protocols that provide Quality of Service. It is also likely that complete new protocols could be designed based on the use of this signalling mechanism.

Regarding SpaceFibre, there is still work to be done regarding the integration of link-layer Quality of Service to the network layer. In particular, the design of a SpaceFibre router with multiple virtual channels supporting Quality of Service is a challenging task.

Looking further into the future, it could even be possible to incorporate the Quality of Service mechanisms of SpaceFibre into a new version of SpaceWire. The biggest challenge would be to include all of these new capabilities while keeping compatibility with components designed with the original SpaceWire specification.

8.6 Conclusions

This research has shown that the development of a Quality of Service layer for SpaceWire needs to take into account the specific characteristics of wormhole switching, the constraints imposed by the limitations of space-qualified electronics and the advantages of a much more controlled network setting. This is in contrast with the generic network traffic targeted by Quality of Service solutions available for commercial ground applications.

Spacecraft avionics, in particular data handling systems, can take advantage of this controlled network setting and chose the specific protocol that best suits the user's needs and provides the best results. This is better than using an all-in-one solution that it is not so good for the particular scenario considered. This thesis follows this recommendation and therefore it provides different protocols with Quality of Service for the types of use cases described in Table 8-1.

Table 8-1: Recommended developed protocols depending on the use case

Use case	Recommended protocol
Asynchronous network with bidirectional or unidirectional data flows that require reliability and end to end flow control.	BTP protocol in asynchronous mode with segmentation. An upper bound for the latency can be computed analytically.
Asynchronous network or point to point connections with unidirectional high data-rate data flows that require reliability.	LBP protocol can be used instead of BTP protocol for increased performance using large SpaceWire packets.
Synchronous network with bidirectional data flows that require reliability, end to end flow control and deterministic behaviour.	BTP protocol in synchronous mode.
Synchronous network with unidirectional data flows that require reliability, end to end flow control, deterministic behaviour and strict requirements on the maximum latency for command and control operations.	UTP protocol.
Synchronous network with traffic generated sporadically or randomly with strict requirements on latency and throughput for high priority data flows.	NAP protocol in conjunction with the LBP protocol if reliability is also required.
Synchronous network that uses RMAP protocol and require reliability, determinism, and strict requirements on the maximum latency for command and control operations.	RMAP scheduler.
Network that requires gigabit data rates links with configurable selection of the Quality Of Service type desired.	SpaceFibre technology with integrated Quality of Service at link layer.

However, the characteristics of most payload data handling systems lead to some protocols being more likely to be used because they cover most typical use cases. This applies especially for SpaceFibre and the RMAP scheduler (i.e. SpaceWire-D), which are in the process of being standardised by ESA and the international community. This thesis has made a significant contribution to the definition, prototyping and validation of these protocols with an in-depth analysis of its capabilities including hardware prototyping in FPGA technology.

Appendix A: Publications Related to This Work

Based on the work described in this thesis the following publications have been produced:

- Parkes S., Ferrer-Florit A., "SpaceWire-RT Initial Protocol Definition", Draft 1.0, SpaceNet Report No. SpW-RT WP3-200.1, ESA Contract No. 220774-07-NL/LvH, March 2008.

- Ferrer-Florit A., Parkes S., "SpaceWire-RT", International SpaceWire Conference, Nara, Japan, November 2008, pp. 15-23.

- Ferrer-Florit A., Parkes S., "SpaceWire-RT Prototyping", International SpaceWire Conference, Nara, Japan, November 2008, pp. 113-120.

- Ferrer-Florit A., Parkes S., Mendham P., "Quality of Service in NoC for Reconfigurable Space Applications", AHS pp.482-487, 2009 NASA/ESA Conference on Adaptive Hardware and Systems, San Francisco, USA, July 2009, pp. 482-487.

- Ferrer-Florit A., Parkes S., "Unified Communication Infrastructure for Small Satellites", International Astronautical Congress, October 2009 (IAC-09.B4.6A.1), pp. 3776-3780.

- Parkes S., Ferrer-Florit A., "SpaceWire-D Deterministic Control and Data Delivery Over SpaceWire Networks", ESA Contract No. 220774-07-NL/LvH, University of Dundee, April 2010.

- Ferrer-Florit A., Parkes S., "SpaceWire-D Prototyping", International SpaceWire Conference, St Petersburg, Russia, June 2010, pp. 391-395.

- Parkes S., Ferrer-Florit A., "SpaceWire-D: Deterministic Data Delivery with SpaceWire" International SpaceWire Conference, St Petersburg, Russia, June 2010, pp. 31-39.

- Ferrer-Florit A., Parkes S., Gonzalez-Villafranca A., Suess M., "Hardware Implementation of an RMAP network scheduler", International SpaceWire Conference, San Antonio, USA, November 2011, pp. 121-128.

- Suess M., Ferrer-Florit A., "Avoiding SpaceWire Network Congestion", International SpaceWire Conference, San Antonio, USA, November 2011, pp. 129-133.

- S. Parkes, A. Ferrer, A. Gonzalez, & C. McClements, "SpaceFibre Standard Draft E1", University of Dundee, 28th September 2012.

- Parkes S., Ferrer-Florit A., Gonzalez-Villafranca A., McClements C., "SpaceFibre: Multiple Gbit/s Network Technology with QoS, FDIR and SpaceWire packet transfer capabilities", International SpaceWire Conference, Gothenburg, Sweden, June 2013, pp. 11-18.

- Parkes S., Ferrer-Florit A., Gonzalez-Villafranca A., McClements C., Ginosar R., Liran T., Alon D., Goldberg M., Sokolov G., Burdo G., Blatt N., Rastetter P., Krstic M., Crescenzio A., "A Radiation Tolerant SpaceFibre Interface Device", International SpaceWire Conference, Gothenburg, Sweden, June 2013, pp. 123-128.

- Ferrer-Florit A., Gonzalez-Villafranca A., McClements C., Parkes S., "STAR Fire: SpaceFibre Diagnostic Interface and Analyser", International SpaceWire Conference, Gothenburg, Sweden, June 2013, pp. 290-293.

Appendix B: SpaceWire Devices

This appendix presents the main SpaceWire devices used in the experiments described in this thesis.

B.1 SpaceWire-10X Router

The SpW-10X router is a fully compliant SpaceWire routing switch device that implements a non-blocking switch using wormhole and group-adaptive routing. It supports Time-code distribution and RMAP protocol for device configuration [McClements 2008].

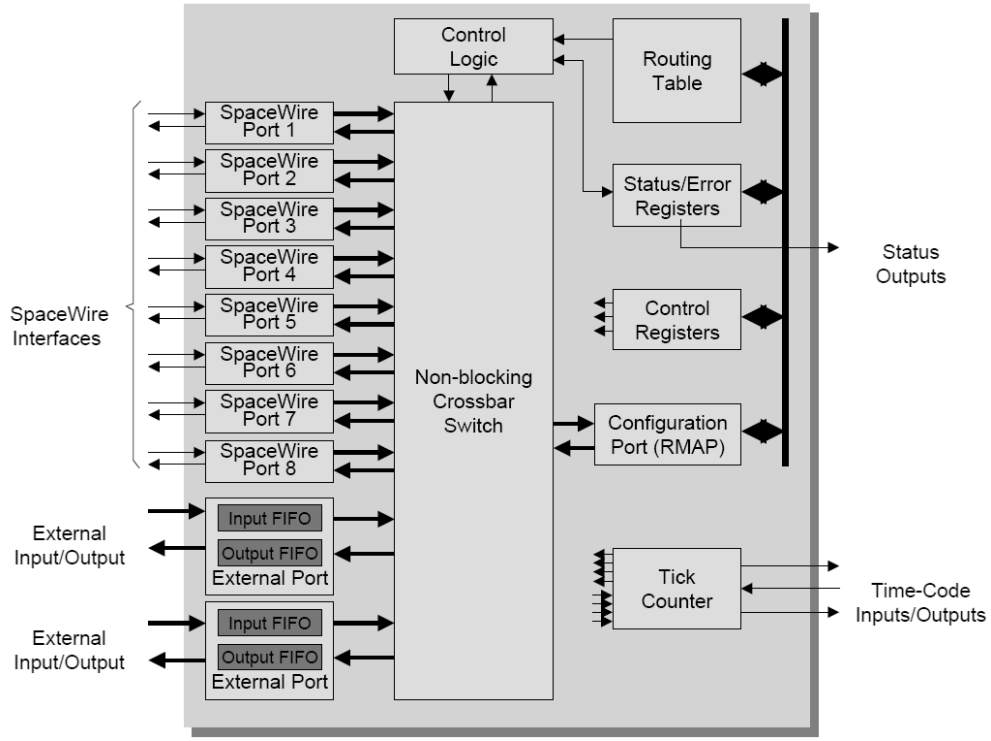


Figure B-1: SpaceWire 10X block diagram [ATMEL 2010c]

The SpW-10X has eight SpaceWire ports, two external ports and one configuration port. Once a SpaceWire packet arrives at an input port, its first byte is used to route the whole packet until an EOP or EEP maker is found. Depending on the value of the first byte, path addressing or logical addressing is used. The router contains a routing table that enables

the use of adaptive routing and two different priority levels within a round-robin arbitration. Figure B-1 shows the block diagram with the crossbar switch.

When a packet requests an output port and this destination or output port is currently occupied by another packet, the crossbar connection cannot be provided and the packet requesting this output port becomes blocked or stalled. SpW-10X implements a timeout mechanism that drops blocked packets after a certain timeout expires but it does not retransmit them later [Kleinrock 1997] which should be done by a higher-level protocol.

B.2 SpaceWire Remote Terminal Controller

The SpaceWire Remote Terminal Controller (RTC) device is a processor-based SpaceWire node implemented as a System-On-Chip. It provides enough autonomy and capabilities to remote spacecraft terminals and instruments to relieve the central processing chain from repetitive standard acquisitions and management duties.

The device includes an embedded LEON 2 microprocessor, a CAN bus controller, ADC/DAC interfaces for analogue acquisition/conversion, and standard interfaces and resources (UARTs, timers, FIFO and GPIO) [SAAB 2008]. Figure B-2 shows the block diagram.

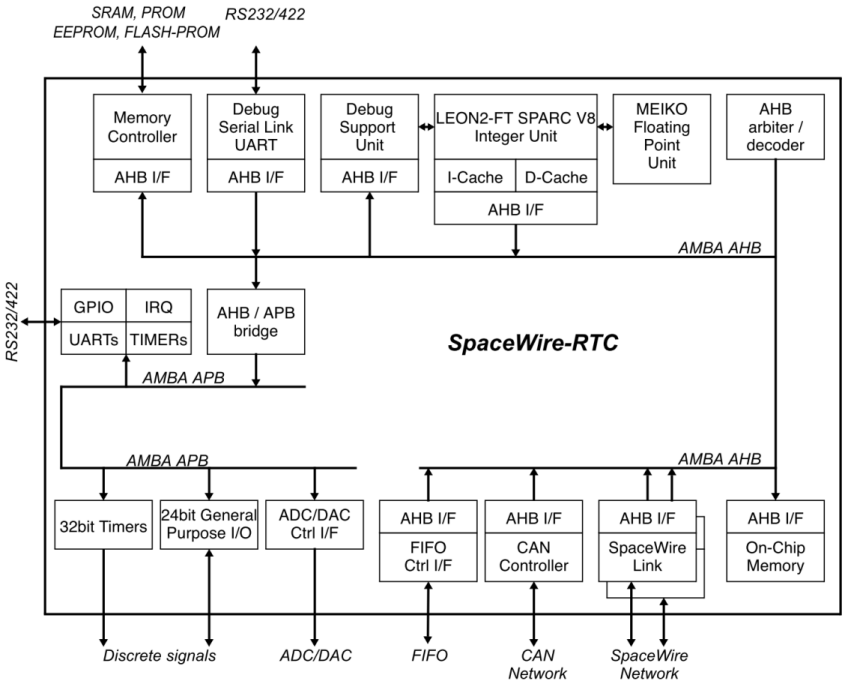


Figure B-2: SpaceWire Remote Terminal Controller block diagram [ATMEL 2010]

B.3 SpaceWire EGSE Equipment

Electrical Ground Support Equipment (EGSE) are commercial units used for ground space application development and testing of space related technologies. The two main units used in this thesis for experimenting with SpaceWire are the STAR-Dundee SpaceWire Brick and the STAR-Dundee Link Analyser [Parkes 2003].

STAR-Dundee SpaceWire Brick is a device that implements a SpaceWire router with three ports, one of them being in fact a USB connection with a computer. It can also be configured to act as a SpaceWire interface with two links. It is typically used as a SpaceWire node, using the SpaceWire USB drivers for Windows and Linux PCs that enables device configuration and SpaceWire data transmission and receiving.

STAR-Dundee link analyser is used for SpaceWire link monitoring. It provides graphical views of the bit level, character level and packet level of SpaceWire standard. It is easy to use, powerful and has plenty of useful tools, such as the error injection option.

References

[Allman 1999] Allman M., Paxson V., Stevens W., “TCP Congestion Control”, Request for Comments 2581, Network Working Group, April 1999.

[ANSI 1993] X3T9.3 Task Group of ANSI, "Fibre Channel Physical and Signaling Interface (FC-PH)", Rev. 4.2, October 1993.

[ARINC 1993] Aeronautical Radio Inc., "MD. ARINC Specification 659: Back- plane Data Bus", Annapolis, December 1993.

[ARINC 2005] Aeronautical Radio Inc., "Aircraft Data Network Part 7 Avionics Full-Duplex Switched Ethernet (AFDX) Network", ARINC Specification 664 Part 7, June 2005.

[ATMEL 2010] Atmel, SpaceWire Remote Terminal Controller (RTC) datasheet, available at www.atmel.com.

[ATMEL 2010b] Atmel, Rad-Hard 32 bit SPARC V8 Processor (AT697F) datasheet, available at www.atmel.com.

[ATMEL 2010c] Atmel, SpW-10X SpaceWire Router (AT7910E) datasheet, available at www.atmel.com.

[Avasare 2000] Avasare P. et al, “Centralized end-to-end flow control in a best-effort Network-On-Chip”, in Proceedings of the 5th ACM international conference on Embedded software, New York, 2005, pp. 17-20.

[Bainbridge 2002] Bainbridge J., Furber S., "Chain: A delay-insensitive chip area interconnect", IEEE Micro, 2002, pp. 16-23.

[Balakrishnan 2001] Balakrishnan H., Seshan S., “The Congestion Manager”, Request for Comments 3124, Network Working Group, June 2001.

[Bertozzi 2004] Bertozzi D., Benini L., “Xpipes: A Network-On-Chip architecture for gigascale systems-on-chip”, IEEE Circuits and Systems Magazine, vol. 4, Issue 2, 2004. pp. 18-31.

[Blahut 1994] Blahut Richard E., Blahut., "Algebraic Codes for Data Transmission", Cambridge University Press, 2002.

[Bolotin 2004] Bolotin E., et al, "QNoC: QoS architecture and design process for network on chip," Journal of Systems Architecture 50, no. 2-3, 2004, pp. 105-128.

[**Borky 1996]** Borky J., Singaraju, B., Stevens, K., "An advanced spacecraft avionics architecture", Aerospace Applications Conference, IEEE Proceedings, 1996, pp. 227-242.

[Budruk 2003] Budruk R, Anderson D, Shanley T, Addison-Wadley, "PCI Express System Architecture", Addison-Wesley, 2003.

[CCSDS 2002] Consultative Committee for Space Data Systems, "Time Code Formats", CCSDS 301.0-B-3 Blue Book, January 2002.

[CCSDS 2007] Consultative Committee for Space Data Systems, "Spacecraft Onboard Interface Services – Informational Report", CCSDS 850.0- G-1 Green Book, Issue 1.0, Washington D.C, June 2007.

[Charlery 2003] Andriahantenaina A., Charlery H., Greiner A., Mortiez L., Zeferino C., "SPIN 2003: a Scalable, Packet Switched, On-Chip Micro- network", Proceedings of the Design Automation and Test in Europe Conference, Embedded Software Forum, March 2003, pp. 70-73.

[Dally 1992] Dally W., "Virtual-channel flow control", IEEE Transactions on Parallel and Distributed Systems, 1992, pp. 194-205.

[Dally 2003] Dally W., Towles B., "Principles and Practices of Interconnection Networks", San Francisco, Morgan Kaufmann Publishers Inc., 2003.

[Dean 2008] Dean B., Warren R., Boyes B., "RMAP over SpaceWire on the ExoMars Rover for Direct Memory Access by Instruments to Mass Memory", 2nd International SpaceWire Conference, November 2008.

[Dielissen 2003] Dielissen J., Radulescu A., Goossens K., Rijpkema E., "Concepts and implementation of the philips network-on-chip", IP-SOC Workshop, November 2003.

[Dier 2002] Dier C., "Radiation effects on spacecraft & aircraft", ESA SP-477, 2002, pp. 505-512,.

[DOD 1975] United States Department of Defense, "MIL-STD-1553B Specification", 1975.

[ECSS 2003] European Cooperation for Space Standardization, "Ground systems and operations Telemetry and telecommand packet", ECSS Standard E-70-41A, January 2003.

[ECSS 2003b] European Cooperation for Space Standardization, "SpaceWire, Links, Nodes, Routers and Networks", ECSS-E-50-12A Issue 1, January 2003.

[ECSS 2008] European Cooperation for Space Standardization, "SpaceWire, Links, Nodes, Routers and Networks", ECSS-E-ST-50-12C Issue 1, July 2008.

[ECSS 2010] European Cooperation for Space Standardization, "SpaceWire protocol identification", ECSS-E-ST-50-51C Issue 1, February 2010.

[ECSS 2010b] European Cooperation for Space Standardization, "SpaceWire - Remote memory access protocol", ECSS-E-ST-50-52C, Feb 2010.

[ESA 2004] European Space Agency (ESA), "EarthCARE – Earth Clouds, Aerosols and Radiation Explorer Technical and Programmatic", Annex to ESA SP-1279(1), April 2004.

[Fattah 2009] Fattah H., Alnuweiri H., "A cross-layer design for dynamic resource block allocation in 3G Long Term Evolution system", IEEE International Conference on MASS, October 2009, pp. 929-934.

[Ferrer 2010] Ferrer A., "SpW-RT as SW on the SpW-Remote Terminal Controller - Demonstration", University of Dundee, Presentation to 14th SpaceWire Working Group Meeting, Noordwijk, The Netherlands, February 2010.

[Ferrer 2010b] Ferrer A, "SpW-D preliminary Protocol Implementation", University of Dundee, Presentation to 15th SpaceWire Working Group Meeting, Noordwijk, The Netherlands, October 2010.

[Ferrer 2013] Ferrer A., Gonzalez-Villafranca A., McClements C., Parkes S., "STAR Fire: SpaceFibre Diagnostic Interface and Analyser", International SpaceWire Conference, Gothenburg, Sweden, June 2013, pp. 290-293.

[Flexray 2006] Flexray Consortium, "FlexRay Communications System Protocol Specification", Version 2.1 Revision A, 2006.

[Gardner 2011] Gardner M., Hunt R., et al, "Joint Architecture Standard (JAS) Reliable Data Delivery Protocol (RDDP) Specification", Sandia National Laboratories, Albuquerque, USA, May 2011.

[Gerla 1996] Gerla M., et al., "Quality of service support in high-speed, wormhole routing networks", International Conference on Network Protocols, 1996, pp. 40-47.

[Gozdecki 2003] Gozdecki J., "Quality of Service Terminology in IP Networks", IEEE Commun. Mag., 2003, pp.153 -159.

[GSFC 2005] NASA Goddard Space Flight Center, "Geostationary Operational Environmental Satellite (GOES), GOES-R Series, GOES-R Reliable Data Delivery Protocol (GRDDP)", 417-R-RPT-0050, Baseline Version 2.1, July 2005.

[Gwaltney 2006] Gwaltney D.A., Briscoe J.M., "Comparison of Communication Architectures for Spacecraft Modular Avionics Systems", Marshall Space Flight Center (NASA), Alabama, June 2006.

[Habinc 2007] Habinc S., Engström K., "SpaceWire Remote Terminal Controller, User Manual, RTC-100-0012", Version 1.7, February 2007.

[Habinc 2010] Habinc S., et al, "Leveraging the Availability of 32-bit Fault-Tolerant Processors Suitable for Radiation-Tolerant FPGA Devices", CMSE, 2010

[Hegarty 2005] Hegarty M., "Avionics Networking Technology", Data Device Corporation (DDC), November 2005

[Hjortnaes 2011] Hjortnaes K., "Introduction and Status of SAVOIR", ESA Workshop on Avionics Data, Control and Software Systems (ADCSS), December 2011.

[Hult 2011] Hult T., "Examples of buildings blocks (CDMU, RTU spec's)", ESA Workshop on Avionics Data, Control and Software Systems (ADCSS), December 2011.

[InfiniBand 2000] InfiniBand Trade Association, "InfiniBand Architecture Specification", Volume 1, Release 1.0, October 2000.

[ISO 2003] International Organization for Standardization, "Controller area network (CAN) Part 1: Data link layer and physical signalling", 2003.

[ITU 1994], International Telecommunication Union, "Open Systems Interconnection - Basic Reference Model: The basic model", Recommendation X.200, July 1994.

[Kamik 2004] T. Kamik, P. Hazucha, and J. Patel, "Characterization of soft errors caused by single event upsets in CMOS processes," IEEE Truns. Dependable and Secure Compuling, Vol. 1, Issue 2, April-June 2004, pp. 128-143.

[Keutzer 2000] Keutzer K., et al, "System-level design: Orthogonalization of concerns and platform-based design", IEEE Trans. on CAD of Integrated Circuits and Systems, 2000, pp. 1523-1543.

[Kleinrock 1997] Kleinrock L., Hu P., "A dynamic timeout scheme for wormhole routing networks", Proceedings of the IEEE International Conference on Communications, 1997, pp. 1406-1410.

[Kopetz 1994] Kopetz H., Gruensteidl G., "TTP - A Protocol for Fault-Tolerant Real-Time Systems", IEEE Computer, Vol. 24 (1), 1994, pp. 14-23.

[Liu 2004] Liu Q., Zhou S., Giannakis G., "Cross-layer modeling of adaptive wireless links for QoS support in multimedia networks", Proc. 1st Int. Conf. QShine, 2004., pp. 65-75.

[Liu 2005] Liu Q., Zhou S., Giannakis G., "Cross-Layer Scheduling With Prescribed QoS Guarantees in Adaptive Wireless Networks," IEEE Journal on Selected Areas in Communications v. 23 no. 5, May 2005, pp. 1056-66.

[McClements 2008] McClements C., Parkes S., "SpW-10X SpaceWire Router AT7910E user manual", Atmel, Issue 3.4, July 2008.

[McKinley 1993] McKinley P. K., Ni L. M., "A survey of wormhole routing techniques in direct networks", IEEE Computer Volume 26 (2), 1993, pp. 62-76.

[Milojevic 2008] Leroy A., Milojevic D., Verkest D., Robert F., "Catthoor: Concepts and Implementation of Spatial Division Multiplexing for Guaranteed Throughput in Networks-on-Chip", IEEE Trans. Computers 57(9), 2008, pp. 1182-1195.

[Nomachi 2010] Nomachi M., "SpaceFiber in Japan", Mitshubishi electric, Presentation to 14th SpaceWire Working Group Meeting, Noordwijk, The Netherlands, February 2010.

[Notebaert 2008] Notebaert O., Gunes-Lasnet S., Farges P. Y., "ECSS and SOIS Standard Services for Communications over a 1553B Bus", Proceedings of DASIA 2008 Data Systems In Aerospace, Noordwijk, Netherlands, 2008.

[Osterloh 2004] B. Osterloh, H. Michalik, B. Fiethe, and K. Kotarowski, "SoCWire: A Network-on-Chip Approach for Reconfigurable System-on-Chip Designs in Space Applications," in NASA/ESA Conference on Adaptive Hardware and Systems, vol. (AHS-2008), (Noordwijk), June 2008, pp. 51-56.

[Parkes 1999] Parkes S., "SpaceWire: The Standard", Proceedings DASIA, Data Systems In Aerospace, May 1999.

[Parkes 2003] Parkes S., McClements C., Mills S., Martin I., "SpaceWire: IP, components, development support and test equipment", DASIA Data Systems in Aerospace, SP-532, Prague, Czech Republic, June 2003.

[Parkes 2006] Parkes S., et al, "Remote Memory Access Protocol", ECSS-E50-11 draft F, December 2006.

[Parkes 2007] Parkes S., McClements C., Dunstan M., "SpaceFibre Outline Specification", University of Dundee, October 2007.

[Parkes 2008] Parkes S., "SpaceWire-RT Requirements", SpaceNet Report No. SpW-RT WP3- 100.1, ESA Contract No. 220774-07-NL/LvH, February 2008.

[Parkes 2008b] Parkes S., “SpaceWire: Spacecraft Onboard Data-Handling Network”, IAC-08.B2.4.1, IAC Glasgow, 2008.

[Parkes 2008c] Parkes S., “SpaceWire-RT Initial Protocol Definition”, Draft 1.0, SpaceNet Report No. SpW-RT WP3-200.1, ESA Contract No. 220774-07-NL/LvH, March 2009.

[Parkes 2008d] Parkes S., Ferrer A., “SpaceWire-RT Initial Protocol Definition”, Draft 1.0, SpaceNet Report No. SpW-RT WP3-200.1, ESA Contract No. 220774-07-NL/LvH, March 2008.

[Parkes 2009] Parkes S., McClements C., Dunstan M., Suess M., “SpaceFibre: Gbit/s Links For Use On board Spacecraft”, International Astronautical Congress, Daejeon, Korea, 2009, paper IAC-09-B2.5.8, pp. 2905-2912.

[Parkes 2009b] Parkes S., "SpaceWire-RT Initial Protocol Definition", Draft 2.1, SpaceNet Report No. SpW-RT WP3-200.1, ESA Contract No. 220774-07-NL/LvH, February 2009.

[Parkes 2010] Parkes S., Ferrer A., “SpaceWire-D: Deterministic Data Delivery with SpaceWire”, International SpaceWire Conference, St Petersburg, Russia, June 2010, pp. 31-39.

[Parkes 2010b] Parkes S., McClements C., Suess M., “SpaceFibre”, International SpaceWire Conference, St Petersburg, Russia, 2010, pp 41-45.

[Parkes 2012] Parkes S., Ferrer A., Gonzalez A., McClements C., “SpaceFibre Standard Draft E1”, University of Dundee, September 2012.

[Parkes 2012b] Parkes S., "SpaceWire User's Guide", STAR-Dundee Limited, 2012.

[RapidIO 2008] RapidIO Trade Association, "RapidIO Specification 2.0", March 2008.

[Rossum 2003] G. Van Rossum. “The Python Language Reference Manual” Network Theory Ltd., September 2003.

[SAAB 2008] Habinc S., Engström K., "SpaceWire Remote Terminal Controller User Manual", RTC-100-0012, Version 1.7, February 2007.

[Saltzer 1984] Saltzer J., Reed D., Clark D., “End-to-end arguments in system design”, ACM Transactions on Computer Systems, November 1984, pp. 277-288.

[Shakkottai 2003] Shakkottai S., et. al., "Cross-layer design for wireless networks", IEEE Commun. Mag., October 2003, pp. 74-80.

[Spitzer 2000] Spitzer, Cary R., "The Avionics HandBook", CRC Press, 1 edition, December 2000.

[STAR-Dundee 2010] STAR-Dundee Ltd, SpaceWire-USB Brick datasheet, available at www.star-dundee.com.

[STAR-Dundee 2010b] STAR-Dundee Ltd, SpaceWire Link Analiser datasheet, available at www.star-dundee.com.

[Sundarapandian 2009] Sundarapandian V., “Probability, Statistics and Queueing Theory", PHI Learning, 2009.

[Tramutola 2011] Tramutola A., Martelli A., Caramia M., "Avionic Architectures for Space Exploration Missions", Proceedings of Data Systems In Aerospace (DASIA), August 2011.

[USB 2008] USB Implementers Forum, "Universal Serial Bus 3.0 Specification", Rev. 1.0, November 2008.

[Villalón 2005] Villalón M., "Estudio de QoS en WLANs IEEE 802.11e", I Congreso Español de Informática (CEDI 2005), Granada, Spain, September 2005.

[VITA 1998] ANSI/VITA, "VME Protocol Specification", Draft Standard VITA 26-199x, Draft 1.1, August 1998.

[Walter 2007] Walter I., et al, “Access regulation to hot-modules in wormhole NoCs”, First IEEE/ACM Int. Symp. on Networks-on-Chip (NOCS07), 2007, pp. 137-148.

[Widmer 1983] Widmer A.X., Franaszek P.A., “A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code”, IBM J. Res. Dev., Vol 27, Issue 5 (440-451), 1983.

[Wiklund 2003] Wiklund D., Dake L., "SoCBUS: switched network on chip for hard real time embedded systems", Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03), 2003, pp. 78-85

[XILINX 2011] Xilinx, Virtex-II Platform FPGA User Guide, available at www.atmel.com.

[XILINX 2011b] Xilinx, Virtex-4 Data Sheet, available at www.atmel.com.

Glossary

Asynchronous network	Event-triggered network in which packets are sent without a synchronization mechanism between nodes.
Babbling idiot	A faulty-node that monopolizes use of a link by sending permanently.
Bandwidth	Used in this thesis as equivalent to maximum throughput of a link.
Bit Error Ratio	Number of bit errors divided by the total number of transferred bits during a studied time interval. It is a unit-less performance measure.
Channel	End-to-end logical connection between two nodes. A channel encapsulates the parameters of an end-to-end connection between two entities located in two different terminals of the network.
Codec	Element capable of encoding or decoding a digital data stream for data communication purposes.
Data integrity	Service that guarantees that data delivered is correct.
D-code	8-bit data symbol of the 8b/10b line code.
Differentiated services	Mechanism that provides different Quality of Service classes to different network traffic types.
End-to-end flow control	Mechanism that regulates the rate of communication between two terminals through a switched network. It ensures that there is always space at the destination buffer before sending a packet.
Epoch	Period of time between the use of two timeslots with the same identifier.
Flow control	Mechanism that allocates network resources to packets as they progress along their path or route.

Guaranteed service	Service class that guarantees a certain level of performance on the services they provide as long as the traffic injected complies with a set of restrictions
Jitter	Difference between the maximum and the minimum duration of a communication action.
K-code	Control symbol of the 8b/10b line code
Latency	Duration of a communication action.
Layer	A term used to define one level of hierarchy of functions, as specified by the OSI reference model
Message	User-application data unit encapsulated in one or more packets.
Network path	Sequence of hops of a packet across the network. In SpaceWire it is determined by the path address field of a packet.
Node	A terminal or end point of the network. In SpaceWire a switching element or routing switch is not a node of the network.
Offered traffic	Total traffic injected to the network by the user.
Packet	A formatted unit of data carried by a communication link.
Packet spilling	Action of discarding the contents of a packet that it is being received.
Protocol Data Unit	Defined as a unit of data which is specified in a protocol of a given layer and which consists of protocol-control information and possibly user data of that layer. In the context of this thesis it refers to the unit of data encapsulated in the cargo field of a packet.
Quality of Service	Set of services provided by the network to the demanding network client or application, related with the performance metrics of the network.

Segment	Unit of user data that it is encapsulated in the cargo of a packet when performing segmentation.
Segmentation	Mechanism that divides a user message into multiple segments to be carried in multiple packets.
Sending priorities	Mechanism in which the order of packets sent by a node depends on their priority.
Service Data Unit	Cargo of a Protocol Data Unit. Unit of data that has been passed down from an OSI layer to a lower layer and that has not yet been encapsulated into a protocol data unit (PDU) by the lower layer.
Slot or timeslot	Smallest time-interval of a TDM schedule.
Synchronous network	Network in which there is a synchronization mechanism between each node or terminal.
TDM	Media access scheme in which access to a link is divided into non-intersecting timeslots of fixed length.
Throughput or data rate	Rate at which information is transferred during a specified time period.
Traffic	Data in a network.
VHDL	Hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits
Virtual Channel	Link-layer logical channel with an independent data buffer. Multiple virtual channels can be multiplexed over a physical link.
Wormhole Switching	Flow control mechanism in which forwarding a packet as soon as the header is received without waiting for the entire packet to be received

Acronyms & Abbreviations

ACK	Acknowledgement
BE	Best-Effort
BER	Bit-Error Rate
BFCT	Buffer Flow Control Token
BTP	Bidirectional Transfer Protocol
CAN	Controller Area Network
CCSDS	Consultative Committee for Space Data Standards
CDMU	Command and Data Management Unit
CRC	Cyclic-Redundancy Checks
DLL	Dynamic-Link Library
DP	Data Packets
E2E	End-to-End
EarthCARE	Earth Clouds, Aerosols and Radiation Explorer
ECSS	European Cooperation for Space Standardization
EGSE	Electrical Ground Support Equipment
ESA	European Space Agency
FDIR	Fault Detection Isolation and Recovery
FIFO	First Input First Output
FIT	Failure In Time
FPGA	Field Programmable Gate Array
GAR	Group Adaptive Routing
GT	Guaranteed Throughput
GUI	Graphical User Interface

LBP	Link Backpressure Protocol
MMFU	Mass Memory and Formatting Unit
MTBF	Mean-Time Between Failures
NACK	Negative Acknowledgment
NAP	Network Arbitration Protocol
NoC	Network-On-Chip
OBC	Onboard Computer
PDU	Protocol Data Unit
PLL	Phase-Locked Loop
QoS	Quality of Service
RMAP	Remote Memory Access Protocol
RTC	Remote Terminal Controller
SAVOIR	Space Avionics Open Interface Architecture
SDU	Service Data Unit
SER	Soft-Error Rate
SoC	System-on-Chip
SOIS	Spacecraft Onboard Interface Services
SpaceWire RT	SpaceWire Reliable-Timely
SpFi	SpaceFibre
SpW	SpaceWire
TTP	Time-Triggered Protocol
TDM	Time-Division Multiplexing
USB	Universal Serial Bus
UTP	Unidirectional Transfer Protocol
VC	Virtual Channel